בסיסי נתונים

עקרונות, מודלים ויישומים

רז הייפרמן



ערך יצחק עמיהוד

ספרי לימוד והכשרה במדעי המחשב
הוצאת הוד־עמי בשיתוף המכון לפריון העבודה והייצור

בסיסי נתונים
עקרונות, מודלים ויישומים

מוקדש באהבה ובהוקרה
לאמי

בסיסי נתונים

עקרונות, מודלים ויישומים

מאת רז הייפרמן

ערך יצחק עמיהוד

ספרי לימוד והכשרה במדעי המחשב

הוצאת הוד־עמי בשיתוף המכון לפריון העבודה והייצור

DATA BASES

Concepts, Models and Applications

by **Raz Heiferman**
edited by **Itzhak Amihud**

הוצאת הוד־עמי
לספרי מחשבים
ת.ד. 560 רמת־גן 52 105

Hod-Ami
P.O.Box 560, Ramat-Gan

הודפס בישראל
תמוז תשמ״ד, יולי 1984

ISRAEL
July, 1984

הקדמה

אחד הכלים לניהול יעיל של ארגונים עסקיים ואחרים הוא השימוש הנכון **בנתונים** (Data) **ובמידע** (Information) המופק מנתונים אלו. השימוש בנתונים ובמידע בתהליך קבלת ההחלטות היה קיים, כמובן, גם בעבר, לרוב לא באופן מודע ונחשב יותר ל"אינטואיציה" או ל"חוש טוב לעסקים".

ההתפתחות של המחשב האלקטרוני בתחום החומרה ותוכנת מערכות ההפעלה היתה, והינה מלווה, בהתפתחותם של כלי עזר לניהול נתונים. כלים אלו הולכים ומשתכללים כל העת ומטרתם להפוך את הנתונים והמידע המופק מהם לאחד המשאבים (Resources) העומדים לרשותו של המנהל. למשאבים הקלסיים המקובלים של עבודה, הון ונכסים מצטרף מעתה המשאב הנוסף של **המידע**. כלים משופרים אלו נקראים בשם **הטכנולוגיה של בסיסי הנתונים** (Data Base Technology), ובהם דן הספר שלפנינו.

אנו עדים כל הזמן להופעת אוסף של עקרונות, כלים וטכניקות אשר מטרתם לאפשר ניהול כמויות עצומות של נתונים בעלי מבנה וקשרים מורכבים מאד. כל אלה מרכיבים מערכות תוכנה מיוחדות — **מערכות לניהול בסיסי נתונים** (DBMS — Data Base Management Systems). הופעתן ומידת השימוש בהן נבעו מן הצורך להקים מערכות מידע ממוחשבות ולהרחיב כל העת את שירותיהן.

עם הופעת מערכות אלה האמינו הארגונים שזאת התשובה לבעיות עיבוד הנתונים. במהרה התברר שהציפיות וההכרזות אודות טכנולוגיה זו לא עלו בקנה אחד עם יכולתה. התברר שמערכות אלה לא פתרו את כל הבעיות שהיו ידועות, ואף הציבו בעיות חדשות. עם זאת, למרות המגרעות של טכולוגיית בסיסי־הנתונים, אין היום ספק שהיא מהווה קפיצה קדימה באפשרויות ניהול הנתונים לעומת הטכנולוגיה שקדמה לה. היום ניתן למצוא מערכות לניהול בסיסי נתונים ברוב הארגונים העיסקיים הגדולים אשר מפעילים מערכות מידע מורכבות, במערכות תיב"ם (תכנון ייצור באמצעות מחשב), במערכות לתמיכה בקבלת החלטות, במערכות וידאו־טקס ועוד.

מטרת ספר זה להציג בפני הקורא את העקרונות של טכנולוגיית בסיסי הנתונים ואת הצורה שבה עקרונות אלה יושמו במציאות באמצעות המערכות לניהול בסיסי נתונים. הספר נועד כספר לימוד, בהמשך לנושא ארגון נתונים וניהול קבצים, וכמקור ללימוד עצמי לכל מי שעוסק במערכות מידע ומעוניין להכיר ולהבין טוב יותר את התפיסה העומדת מאחורי טכנולוגיה זו.

הספר מחולק לשני חלקים:

החלק הראשון עוסק בצד העקרוני ומציג את התפיסה והרעיונות המרכזיים של המערכות לניהול בסיסי נתונים. מוצגים המודלים השונים של הנתונים תוך דיון מעמיק בכל אחד מהמודלים המוכרים והשימושיים.

הפרק הראשון מציג את גישת בסיס הנתונים מול הגישה המקובלת לניהול קבצים ומסביר את המושג המרכזי של גישה זו, אי התלות בנתונים.

הפרק השני מהווה סקירה קצרה של הטכניקות לאחסון וניהול קבצים אשר מהוות בסיס להבנת עקרונות הפעולה של המערכות לניהול בסיסי נתונים. הקורא המעונין להעמיק ולבסס את ידיעותיו בנושא זה מופנה אל הספר "ארגון נתונים וניהול קבצים" מאת המחבר בהוצאת הוד־עמי, 1982.

הפרק השלישי מציג את מודל הנתונים ואת מושגי היסוד הדרושים להבנת נושא חשוב זה.

הפרק הרביעי מציג מודל כללי של מערכת לניהול בסיסי נתונים ומסביר מושגים כגון סכימה, תת־סכימה, שפה להגדרת נתונים, שפה לטיפול בנתונים, ואת עקרונות הפעולה של מערכות אלה.

הפרקים החמישי, השישי והשביעי כוללים דיון מעמיק יותר בכל אחד משלשת המודלים העיקריים של הנתונים.

הפרק השמיני מציג את נושא תכנון מודל הנתונים ומציג לקורא את הטכניקה של תיקנון (Normalization) מבנה הנתונים.

הפרק התשיעי מתאר בקצרה את נושא הבטיחות ושלימות הנתונים במערכת לניהול בסיסי נתונים.

הפרק העשירי מציג מספר מגמות וכיוונים עתידיים אפשריים לטכנולוגיה זו.

בחלק השני מוצגות מספר מערכות מסחריות המהוות יישום של הרעיונות שהוצגו בחלק הראשון. אין הספר מתיימר להציג מערכת כלשהי בצורה שלימה ומלאה. המטרה היא לאפשר לקורא לרכוש הבנה נוספת במערכות לניהול בסיסי נתונים על־ידי הכרת מערכות נפוצות. המערכות IMS, IDMS, TOTAL, ADABAS, SQL/DS מוצגות כשיטות ליישום של מודל נתונים מתוך אלו שנסקרו. התיאור מבוסס על המידע הרשמי המופץ לציבור על־ידי יצרניות התוכנה.

בסיום הספר ניתנים נספחים הכוללים טבלה המציגה נתונים בסיסיים עבור רוב המערכות הקיימות לניהול בסיסי נתונים, אינדקס, ביבליוגרפיה ומילון נתונים.

תקוותי שספר זה ימלא את החסר בספרות המקצועית בשפה העברית ויתרום לקהיליית המשתמשים במחשבים, לסטודנט ולתלמיד ולכל מי שמעונין בנושא חשוב זה. הספר מבוסס על נסיון שרכשתי במשך שנות עבודתי כמנתח מערכות ביישום מערכות מידע גדולות באמצעות טכנולוגיה זו. כמו כן מבוסס הספר על רשימות הרצאותי בקורסים לניהול בסיסי נתונים במכון לפריון העבודה והייצור ובטכניון בתל־אביב. תודתי לכל התלמידים אשר העירו את הערותיהם ותרמו בצורה זו או אחרת לעיצוב ספר זה. לא אוכל לסיים הקדמה זו ללא תודה מיוחדת לאשתי אשר עודדה אותי ועזרה לי במשך כל התקופה הארוכה שבה כתבתי את הספר.

תוכן העניינים

חלק ראשון: עקרונות של מערכות לניהול בסיסי נתונים

פרק א: גישת בסיס הנתונים

פרק ב: אחסון נתונים וארגון קבצים

פרק ג: מודל הנתונים

פרק ד: מערכות לניהול בסיסי נתונים — מרכיבים ועקרונות פעולה

פרק י: מערכות DBMS — מגמות וכיוונים לעתיד

חלק שני: מערכות DBMS מסחריות

פרק יא: IMS — Information Management System

פרק יב IDMS — Integrated Database Management System

פרק יג: TOTAL

חלק ראשון

עקרונות של
מערכות לניהול בסיסי נתונים

פרק א

גישת בסיס הנתונים

1. מבוא
2. הגישה המקובלת לניהול נתונים
3. גישת בסיס הנתונים
4. אי־תלות בנתונים
5. ציוני דרך בהתפתחות מערכות לניהול נתונים
6. מנהל בסיס הנתונים
7. סיכום
8. שאלות

1. מבוא

"קופרניקוס, האסטרונום והמתמטיקאי הידוע, פרסם בשנת 1543 את ספרו Concerning the Revolutions of Celestial Spheres שהציג תיאוריה חדשה אודות התנועה הפיסית של כדור הארץ, הכוכבים והשמש. תיאוריה זו היתה מנוגדת לחלוטין לתיאוריה של מרכזיות כדור הארץ, שהיתה מקובלת כבר כ־1400 שנה. התיאוריה של מרכזיות השמש שהוצעה ע"י קופרניקוס התקבלה בביקורת רבה, וספרו של קופרניקוס הוכנס לרשימת הספרים האסורים ונשאר שם כ־200 שנה.

השתמשתי בדוגמא של קופרניקוס כדי להדגים דיעה הרווחת בעולם מערכות המידע. מערכות אלה, וכל החשיבה והתכנון הקשורים להן, היו מבוססים על עקרון מרכזיות המחשב בלבד. כפי שהאנשים הקדומים ראו את השמש כסובבת סביב כדור הארץ, כך ראו אנשי מערכות המידע הראשונים את המחשב וסביבו קובץ סדרתי. כל אחד מהמודלים האלה התאים לזמנו ולמקומו. לאחר זמן מה הוא נמצא כלא נכון והיה צורך להחליפו במודל אחר, אשר ייצג בצורה נכונה יותר את ההתנהגות האמיתית של העולם. קופרניקוס הציג נקודת מבט חדשה והניח את הבסיס למכניקה המודרנית. נקודת מבט זו נתנה לנו הבנה במסתרי תנועת השמש והכוכבים ברקיע. בסיס חדש להבנה קיים היום גם בשטח מערכות המידע והוא מושג על ידי מעבר מנקודת המבט של **מרכזיות המחשב** לנקודת מבט של **מרכזיות בסיס הנתונים**".

עד כאן מובאה (בתרגום חופשי) ממאמרו של Charles W. Bachman, אחד מאבות המערכות לניהול בסיסי נתונים, אשר הוצג בשנת 1973 בעת קבלת הפרס היוקרתי ביותר בתחום המחשבים — ACM Turing Award. המובאה מציגה בצורה בולטת את ההבדל בין מערכות מידע המבוססות על מערכות ניהול נתונים מסורתיות, לבין אלה המבוססות על מערכות לניהול בסיסי נתונים.

בארגון המודרני, בנוסף למשאבים הכלכליים הקלסיים של עבודה, הון ונכסים ממשיים כוללים משאב חשוב נוסף — **הנתונים**. נתונים על פעילויות, מוצרים, מצב פיננסי, מתחרים, וכדומה משמשים את הארגון לתפעול שוטף ולקבלת החלטות. שימוש יעיל בנתונים אשר זמינים במועד ובמידת אמינות גבוהה יכול לעזור לארגון בהשגת מטרותיו. לאחר הופעת המחשבים ה"מסחריים" הראשונים, בתחילת שנות ה־50, הבין עולם העסקים את יכולתם לייעל ולשפר את התהליכים של זרימת ועיבוד הנתונים ובכך לתרום לפעילות העסקית.

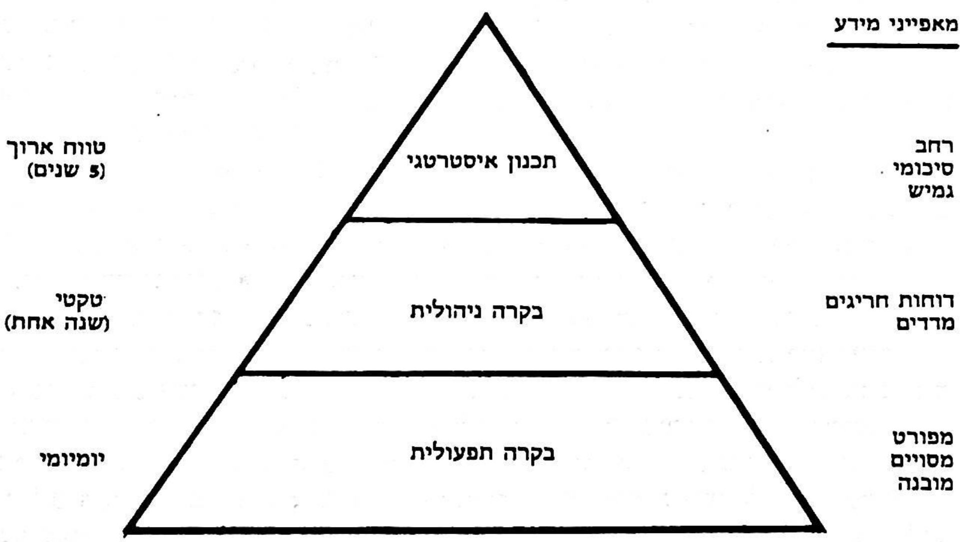
עיבוד נתונים עסקי מחייב את הארגון לאסוף, לצבור ולעבד כמויות גדולות של נתונים. השימוש במחשב מחייב שכמות אדירה זו של נתונים תנוהל בצורה המובנת למחשב, כלומר בצורת רישום מגנטית כלשהי. מערכות המידע הממוחשבות הראשונות השתמשו בקבצים סדרתיים המאוחסנים בסרטים מגנטיים. אפשרות לגישה מהירה יותר לנתונים ובעיקר — גישה באופן ישיר הושגה על ידי ההתקדמות שחלה בשטח החומרה עם הופעת טכנולוגית הדיסק המגנטי. שיפור בשיטות הארגון של הנתונים והגישה אליהם הושג עם ההתקדמות בתוכנה התומכת במערכות המידע ובעיקר על ידי **המערכות לניהול קבצים**. מערכות אלה שחררו את המשתמש מפעולות הקשורות בייצוג הפיסי של הנתונים ביחידות האחסון ומן הטיפול באחזקתם. המערכות לניהול קבצים היוו צעד גדול קדימה בטכנולוגיית עיבוד הנתונים ובכך איפשרו פיתוח מערכות מידע ממוחשבות ביתר קלות.

ריבוי המחשבים והרחבת השימוש בהם במגזר העסקי גרם לצמיחה מהירה מאד בכמות הנתונים המאוחסנים ביחידות האחסון של המחשב. ניהול משאב הנתונים בצורה יעילה הפך לאחד האתגרים הגדולים של הארגון העסקי. במערכות לניהול נתונים התעוררו בעיות אשר

נגרמו עקב התלות הרבה שבין הנתונים לבין תהליכי עיבוד הנתונים. כך ניתן התמריץ לפיתוח טכנולוגיה חדשה לניהול נתונים, **מערכות לניהול בסיסי נתונים** (DBMS—Data Base Management Systems). מערכות אלו מקנות מספר יתרונות על פני צורות ניהול נתונים אחרות והן פותחות בפני הנהלת הארגון דרכים חדשות לניהול ולאחזור מידע עסקי.

המערכות לניהול בסיסי נתונים עדיין אינן כלי נפוץ והן קיימות אצל מיעוט משתמשים, בעיקר הגדולים שבהם, אשר להם מערכות מידע מורכבות אשר פותחו מאמצע שנות ה־70. עם זאת, אין ספק שנהיה עדים למספר הולך וגדל של מערכות מידע המבוססות על בסיסי נתונים. ככל שההכרה בחשיבות הנתונים כמשאב מרכזי בארגון תלך ותגדל, גם השימוש בטכנולוגיית בסיסי הנתונים יהיה נפוץ יותר.

המערכות לניהול בסיסי נתונים כוללות פונקציות שירות רבות ועל כן הן צורכות משאבי מחשב (זכרון, זמן יע"מ) רבים יותר מאשר מערכות התוכנה לניהול קבצים. המחשבים החדשים המגיעים לשוק זולים יותר באופן יחסי ובכך מאפשרים את יישום טכנולוגיית בסיסי הנתונים. במקביל נעשים נסיונות להפעלת מחשבים מיוחדים למתן שרותי בסיס נתונים (DBM—Data Base Machines) באמצעות חומרה, בצידם של המחשבים המרכזיים המבצעים את פעילות עיבוד הנתונים.



תרשים 0 – 1: צרכי המידע בארגון

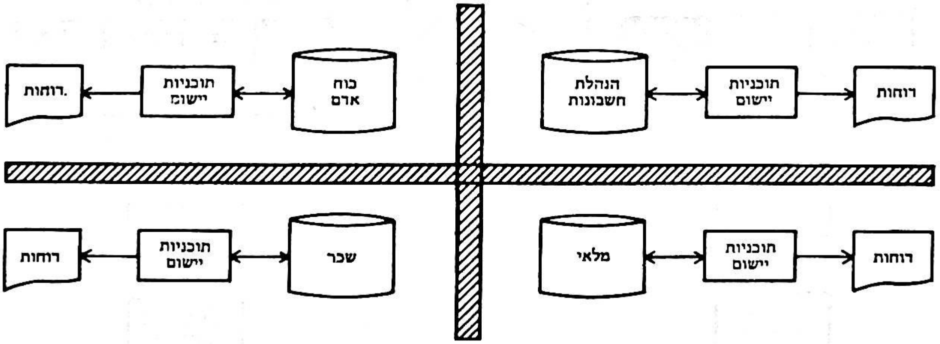
2. הגישה המקובלת לניהול נתונים

החלוקה המקובלת של הארגון העסקי מבחינת תהליכי קבלת החלטות היא לשלש רמות: הרמה התפעולית, הרמה הטקטית והרמה האיסטרטגית. ברמה התפעולית מתקבלות ההחלטות היום־יומיות הקשורות בתפעול שוטף של הארגון. בדרך כלל, תהליכי קבלת החלטות בדרג זה הינם פשוטים יחסית מאחר וכללי ההחלטה הינם ברורים ופשוטים. ברמה הטקטית מתקבלות ההחלטות הקשורות בהקצאת משאבים ותכנון, כגון קביעת תקציבים, ניהול כוח אדם, מדיניות פרסום וכדומה. ההחלטות ברמה זו מורכבות יותר ומבוססות בעיקרן על הערכות לגבי העתיד מבחינת התפתחויות צפויות. ברמה האיסטרטגית מתקבלות ההחלטות מבחינת מדיניות הארגון לטווח הארוך יותר ולכן הן מורכבות עוד יותר.

מבחינת המבנה הארגוני, החלוקה המקובלת של הארגון העסקי היא למחלקות, אגפים, חטיבות וכדומה. חלוקה זו מתאימה לפעילותן העסקית של יחידות ארגוניות אלה. לדוגמא, מחלקת כוח אדם עוסקת בכל ההיבטים הקשורים לניהול כוח אדם (הדרכה, קידום עובדים, תקנים ועוד) בעוד מחלקת הפרסום עוסקת בכל ההיבטים, הקשורים בפעילות הפרסום של הארגון.

מערכות המידע המקובלות חולקו גם הן בהתאם לשתי החלוקות: רמת קבלת החלטות והמבנה הארגוני. ניתן לסווג את רוב מערכות המידע המקובלות כמערכות המשרתות בעיקר את הרמה התפעולית בהתאם לחלוקה הארגונית. מערכות אלה מטפלות בכמויות גדולות של נתונים בתהליכים פשוטים יחסית ומוגדרים היטב. לדוגמא, מערכות להנהלת חשבונות, מערכות ניהול מלאי וכדומה. ההצלחה נובעת בעיקר ממהירות העבודה העצומה של המחשב ויכולתו לטפל ביעילות ובדיוק בכמויות גדולות של נתונים.

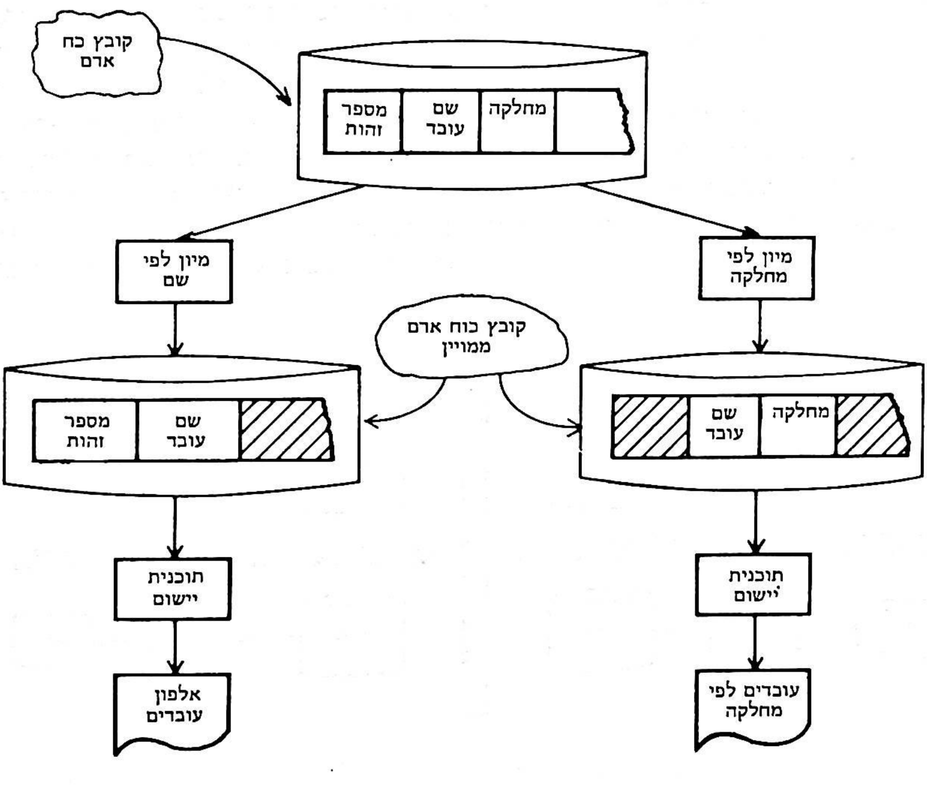
החלוקה בהתאם ליחידות ארגוניות הביאה להקמת קבצים נפרדים לכל יישום בהתאם להשתייכות הארגונית של היחידה שעבורה הוקמה מערכת המידע. תרשים 1־1 מציג את הגבולות שנוצרו בין היישומים השונים באותו ארגון עסקי. לדוגמא, קובץ כוח אדם המשרת את מחלקת כוח אדם בלבד. מחלקת השכר מנהלת קובץ נפרד המשרת אותה בלבד, אך מכיל גם חלק מהנתונים שנמצאים בקובץ כוח האדם.



תרשים 1 – 1: גישת "הקובץ שלי"

החלוקה הארגונית השפיעה על התוכן ועל המבנה של הקובץ, אשר נקבעו בהתאם לדרישות המידע של היחידה הארגונית שאותה משרתת מערכת המידע ובהתאם לשיטות העיבוד. לדוגמא, בקובץ כוח אדם ניתן למצוא נתונים כגון מספר זהות, שם העובד, כתובת, המחלקה שבה הוא מועסק, תפקיד וכדומה, אך לא נמצא בקובץ זה נתונים הקשורים לחישוב משכורת העובד מכיון שהקובץ אינו מיועד למטרה זו. נתונים אלה נמצאים בקובץ השכר שממנו מופקת משכורת העובד.

מערכות המידע חייבות לשאת שינויים מתמידים בדרישות המידע ובנוהלי עיבוד המידע. תוך כדי הפעלת המערכת מתעורר הצורך בנתונים חדשים או בעיבודים חדשים של נתונים קיימים. מבנה הקבצים נוטה להתאים ליישומים המקוריים שעבורם תוכנן אבל אינו מתאים לרוב ליישומים החדשים. קובץ כוח אדם למשל, המשרת את מחלקת כוח אדם מכיל מספר רב של נתונים וממוין לפי מספר זהות העובד שהוא סדר המיון הנוח לרוב תוכניות היישום. כל תוכניות היישום במחלקת כח האדם משתמשות באותו קובץ אך מכיון שאין הן זקוקות לכל הנתונים הכלולים בו, הן מתעלמות מחלק מהם. תרשים 2—1 מציג דוגמא של קובץ כוח אדם אשר משרת שתי תוכניות יישום, אחת — להפקת דוח אלפון עובדים (רשימת עובדים ממויינת לפי שמות העובדים) ואילו השניה — להפקת דוח עובדים לפי המחלקות. כל תוכנית "מתעלמת" מהנתונים שאינם דרושים לה וממיינת את הקובץ לפי סדר המיון המבוקש.

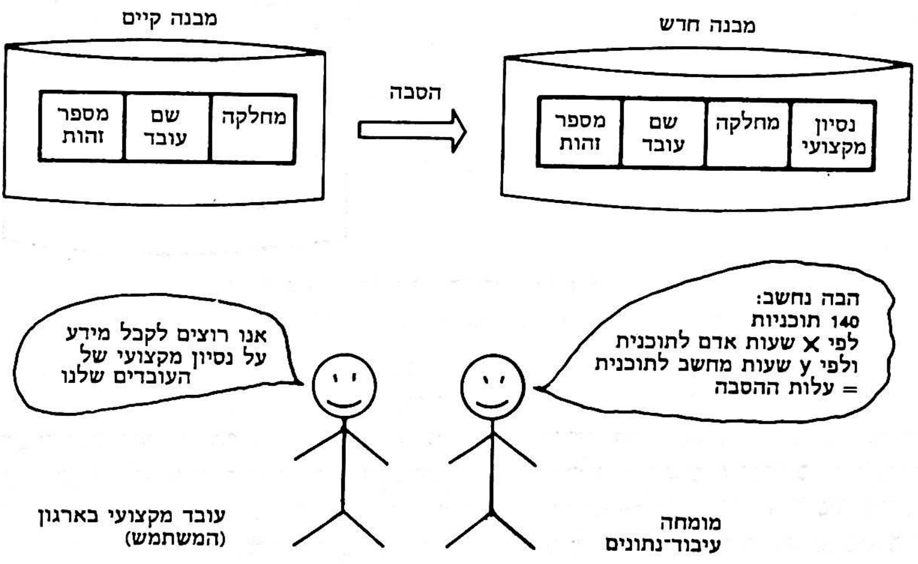


תרשים 2 – 1: רשומה זהה המשמשת תוכניות יישום שונות

נשתמש בדוגמה של הוספת נתונים לקובץ על אודות הנסיון המקצועי הקודם של העובד. קיימות שתי חלופות (אלטרנטיבות) לביצוע השינוי המבוקש.

2.1. הוספת הנתונים החדשים לקובץ

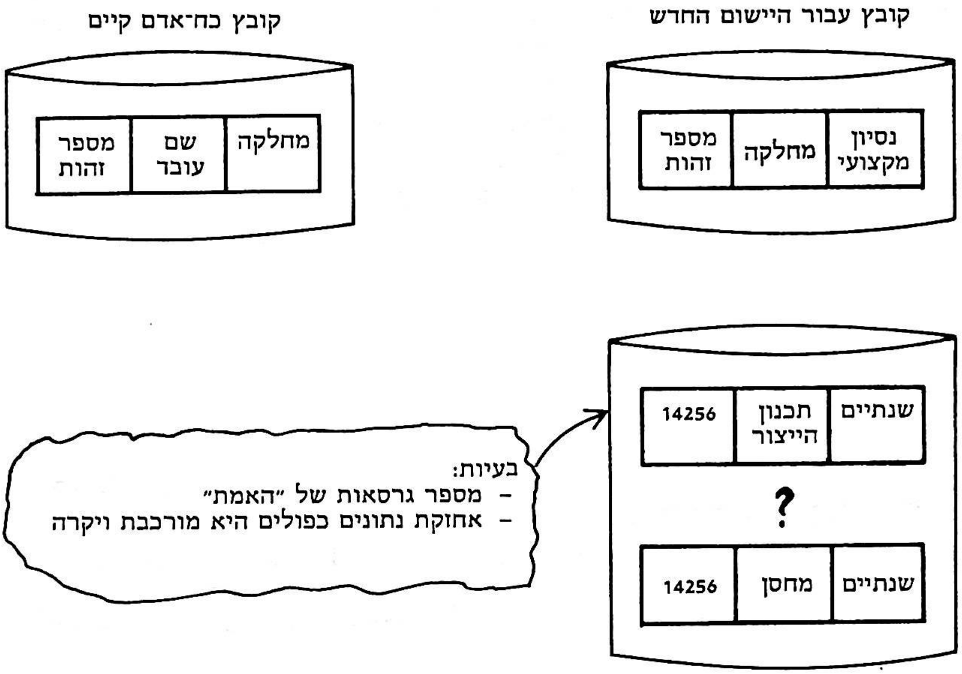
בשיטה זו "מרחיבים" את הרשומה הקיימת כדי שתכיל גם את הנתונים החדשים. מכיוון שמבנה הרשומה משתנה, יש לשנות בהתאם את **כל** תוכניות היישום המשתמשות בקובץ. בדרך כלל יש לבצע הידור (Compilation) מחדש של תוכניות היישום, אך יתכן שיש צורך בביצוע שינויים מורכבים יותר. למרות ששינוי כזה לא נראה מסובך הוא גורר אחריו שינויים בתוכניות יישום רבות המשתמשות בקובץ וכתוצאה — כמות העבודה שיש להשקיע לביצוע השינוי היא רבה. במערכות מידע אלו נוטים להשתמש במבני רשומות לא מעודכנים, או במבני נתונים שאינם מתאימים, מכיוון שדוחים ככל האפשר את הכנסת השינויים הדרושים או שמחליטים שלא לבצע אותם כלל. שינוי מבנה הקובץ ממבנה קיים למבנה חדש קרוי **הסבה** (Conversion).



תרשים 3 - 1: שינוי מבנה הרשומה

2.2. בניית קובץ חדש חלקי

מקימים קובץ חדש אשר מכיל רק את הנתונים הדרושים מתוך הקובץ הקיים ואת הנתונים החדשים. בשיטה זו אין צורך לשנות את כל התוכניות הקיימות ולהסב את מבנה הקובץ. **החסרון של השיטה הוא כפילות הנתונים אשר עלולה לגרום להיווצרות שתי גרסאות של הנתונים כתוצאה של תהליכי עדכון ועיבוד הנפרדים.**



תרשים 4 – 1: פתרון על־ידי יצירת כפילות נתונים

יש לציין שכפילות הנתונים קיימת גם אם נבחר באלטרנטיבה הראשונה. הכפילות נובעת מהעובדה שהחלוקה הפונקציונלית, כלומר אוסף הפעילויות העסקיות בעלות אופי דומה, לא תמיד זהה לחלוקה הארגונית. לדוגמא, נתונים הקשורים למכירות מעניינים את מחלקת השיווק, מחלקת הייצור ומחלקת הנהלת חשבונות. מאחר ואלה יישומים נפרדים הם קשורים לקבצים נפרדים, שבחלקם מכילים נתונים זהים. האחריות לניהול הקבצים נמצאת בידי צוותים מקצועיים שונים, המקבלים נתונים ממקורות שונים ומעדכנים לפי נוהלים שונים. במצב זה עלולה כפילות הנתונים לגרום להיווצרות "מספר גרסאות של האמת", לחוסר אמינות ועקביות של הנתונים ובסופו של דבר — להתמוטטות מערכת המידע. מצב זה מודגם בתרשים 4—1.

במשך הזמן החלו להופיע מערכות מידע אשר תומכות גם בתהליכי קבלת החלטות **ברמה הטקטית**. הן השתמשו בדרך כלל בנתונים המופיעים ביישומים שונים, אך מכיון שכל קובץ היה באחריות צוות מקצועי שונה היה צורך בתיאום עם "בעלי הקובץ". זמן מחשב רב בוזבז לביצוע מעברי מיון ומיזוג לקבלת מבנה נתונים המאפשר הפקת מידע משולב מנתונים שנמצאים בקבצים שונים. בגלל הקושי בהגדרת דרישות המידע לרמה ניהולית זו, והצורך לשלב נתונים ממקורות שונים לשם הפקתו, התעוררו בעיות טכניות בהקמת מערכות מידע אלו

וקצב התפתחותן היה איטי מאד. מערכות מידע התומכות גם בתהליכי קבלת החלטות **ברמה האיסטרטגית** כמעט ולא ניתן למצוא. הקושי בהגדרת הדרישות ובהענות לדרישות מידע מורכבות ביותר בזמן תגובה קצר גרם לכך שהרמה הניהולית הבכירה כמעט ואינה יכולה להסתייע במחשב באופן מלא. השימוש במחשב הוא לקבלת מידע חלקי בלבד.

לסיכום ניתן לומר, שהגישה המקובלת לניהול נתונים הציבה מספר בעיות בפני המשתמשים. חלוקת מערכת המידע של הארגון למספר מערכות מידע, בהתאם לחלוקה הארגונית גרמה לכפילות נתונים רבה ולקשיים רבים בהענות לדרישות מידע אמין בזמן תגובה קצר. הצורך בתיאום הגישה לקובץ עם "בעלי הקובץ" יצרה מצב שבו נתונים היו ב"בעלות" היישום ולא בבעלות הארגון.

3. גישת בסיס הנתונים

גישת **בסיס הנתונים** (Data Base Approach) מהווה **גישה חדשה לניהול נתונים. היא** התפתחה כמענה לבעיות הקיימות בגישה המקובלת ובעקבות המגמות של שילוב קבצי הנתונים למערכת אינטגרטיבית אחת תוך מתן אפשרות לגישה מהירה לנתונים במגוון רחב של חתכים. מטרות הגישה החדשה לניהול נתונים הן צמצום כפילויות הנתונים והגדלת אמינותם וזמינותם, תוך בניית מסגרת שתאפשר הענות מהירה לצרכים המשתנים של עיבוד הנתונים בארגון המודרני.

3.1 הגדרה

מערכת לניהול בסיס נתונים (DBMS — Data Base Management System) הינה מערכת אשר מאפשר גישה בו־זמנית של מספר רב של משתמשים שונים למאגר מרכזי אחד של נתונים (Engles, 1970).

נבחן את המרכיבים העיקריים של ההגדרה.

— **מאגר מרכזי:** ריכוז הנתונים במאגר נתונים אחד כללי הינה אחת הסיבות העיקריות לגישת בסיס הנתונים. גישה זו מתייחסת אל כל הנתונים כשייכים למאגר אחד כללי הקרוי **בסיס הנתונים**, ומנתקת את הקשר שהיה קיים בין מבנה הקובץ לבין מבנה הנתונים הדרוש לתוכניות היישום. **בסיס הנתונים** יוגדר כאוסף של נתונים הקשורים ביניהם תוך כפילות נתונים מבוקרת, ומיועדים לשרת מספר רב של תוכניות יישום. הנתונים מאוחסנים בצורה בלתי תלוייה בתוכניות היישום המשתמשות בנתונים. בגלל הניתוק שבין צורת האחסון לבין תוכניות היישום ניתן לבצע שינויים מסויימים במבנה הנתונים מבלי שתהיה לכך השפעה על תוכניות היישום המשתמשות בהם.

אין זה מעשי תמיד לרכז את כל הנתונים של הארגון במאגר אחד. עם זאת, בניית מאגר נתונים משולב הינו אחד התפקידים החשובים בארגון. על ידי הקמת מאגר מרכזי מוציאים את

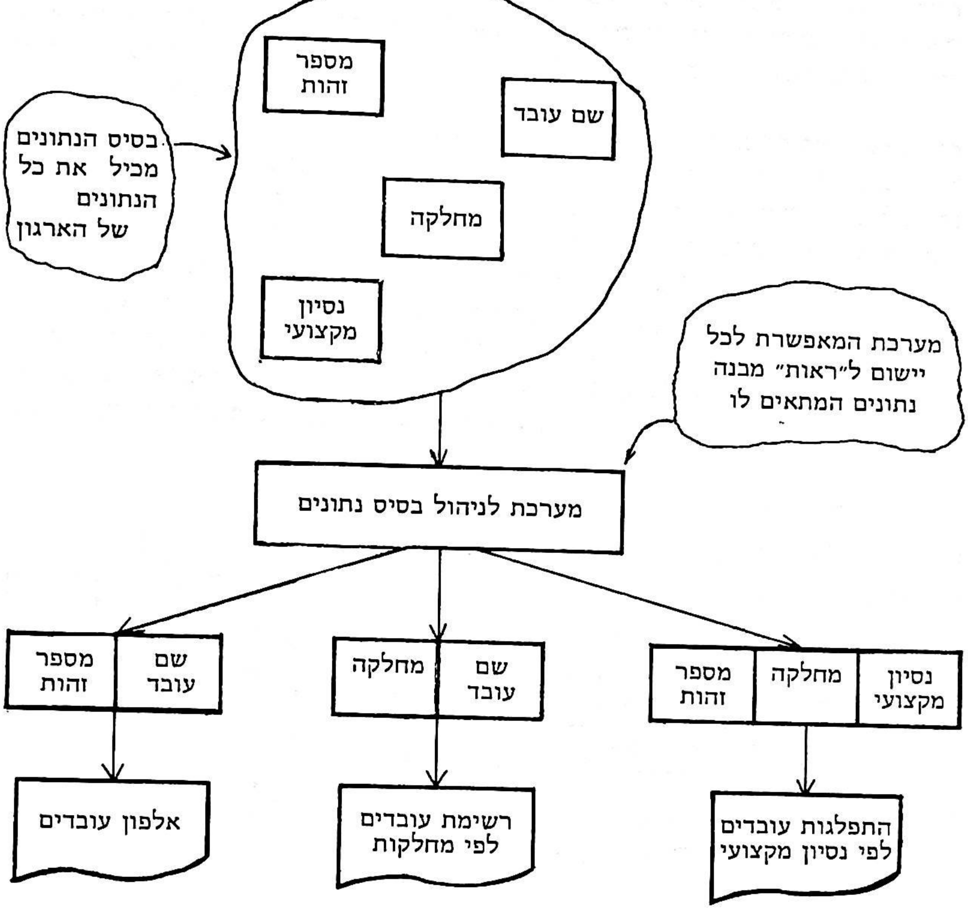
הנתונים מ"בעלות" היישום ומעבירים אותם לפונקציה מרכזית האחראית על ניהול הנתונים, **מנהל בסיס הנתונים** (DBA — Data Base Administrator). ריכוז הנתונים במערכת משולבת אחת עשוי להביא לצמצום בכפילות הנתונים ולהעלות את אמינותם. תהליך הקמת המאגר המרכזי יכול להיות מורכב מאד, ועל־כן מחייב שיתוף פעולה ושיתוף נתונים בין קבוצות שונות של משתמשים שבגישה המקובלת היו אחראים על הנתונים שלהם. כשלון ביצירת שיתוף זה עלולה לגרום לכשלון בהקמת יישומי DBMS. הצלחת גישת בסיס הנתונים מחייבת תיאום מתמיד בין משתמשים שונים ובקרה מרכזית על הנתונים, תפקידים שהוטלו על מנהל בסיס הנתונים.

— **נתונים:** אם בסיס הנתונים צריך לייצג נתונים מועילים אודות העולם המציאותי עליו לייצג אותם כפי שהם קיימים בעולם זה. במשך שנים היו הנתונים מיוצגים בעיקר בהתאם לנוחות העיבוד במחשב, כמו למשל החלוקה ליחידות (רשומות) של 80 תווים בהתאם לגודל כרטיס ניקוב. במציאות הנתונים אינם קיימים ביחידות של 80 תווים אך אפשרויות העיבוד במחשב כפו מבנה זה. הייצוג של העולם המציאותי הינו לעיתים קשה מאד. מערכות ניהול בסיסי הנתונים חייבות להתגבר על מורכבות זו כדי לספק למשתמשים השונים את הנתונים כפי שהם רוצים בהם.

דוגמא למורכבות המציאות הינה הטיפול והעיבוד של נתונים החוזרים על עצמם. ביישום של רכש למשל, ללקוח אחד יכולים להיות מספר רב של הזמנות. אם נתוני הלקוח יחזרו על עצמם כמספר ההזמנות שלו, תהיה בעייה של כפילות הנתונים. המערכת לניהול קבצים מספקות בדרך כלל שני פתרונות: רשומות באורך משתנה, או הפרדה בין קובץ הלקוחות לבין קובץ ההזמנות. שני פתרונות אלה אינם מספקים, ולכן במערכות ניהול בסיסי נתונים אפשריים גם פתרונות אחרים לבעיית ייצוג המציאות.

—**משתמשים שונים:** התכנון של רוב מערכות DBMS מכוון לאפשר למספר רב של משתמשים שונים הפועלים בשטחים פונקציונליים שונים של הארגון להשתמש באותם הנתונים. כלומר, **שיתוף הנתונים** (Data Sharing) הוא מרכיב חשוב במערכת.

— **גישה בו־זמנית:** גישה בו־זמנית לנתונים דרושה כאשר יישומים אחדים הפועלים באצווה עשויים לשלוף ולעדכן נתונים משותפים באותו זמן. ביישומים מקוונים (עיבוד בתקשורת, on-line) נדרשת גישה בו־זמנית לנתונים על ידי משתמשים רבים גם כאשר הם פועלים באותו יישום. הדרישה לאפשר גישה בו־זמנית לנתונים קיימת בכל המערכות המקוונות לסוגיהן השונים. מערכות DBMS מציעות פונקציות מסויימות של נעילה, התאוששות ובטיחות נתונים ובכך הן בעלות יתרון ניכר על פני מערכות רגילות לניהול קבצים בסביבה של תקשורת נתונים.



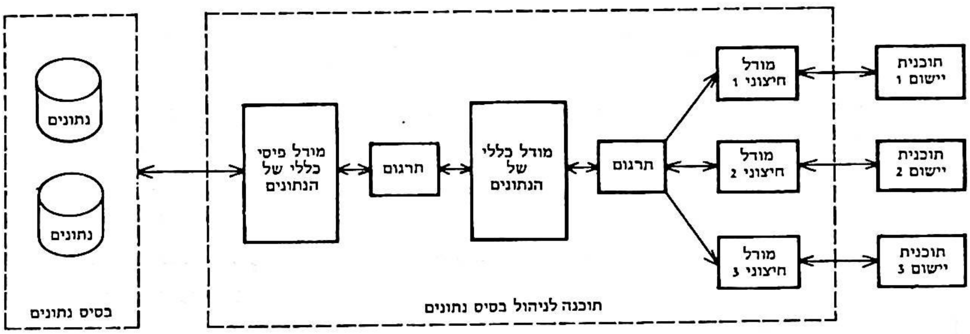
תרשים 5 - 1: התאמת מבנה נתונים לכל יישום

3.2 מודל של מערכת לניהול בסיס נתונים

מערכת לניהול בסיסי נתונים הינה תוכנת מחשב אשר מאפשרת לנהל את מאגר הנתונים ולקיים משטר בטיחות והתאוששות בעת תקלה. הניתוק בין תוכניות היישום לבין מבנה הקובץ מתאפשר על־ידי הגדרת **מודל נתונים כללי** אשר מתאר את כל הנתונים המאוחסנים במאגרי הנתונים, ועל ידי הגדרת **מודלים חיצוניים ייחודיים** אשר נותנים לכל משתמש או תוכנית יישום את **תמונת הנתונים** שהם זקוקים להם.

הקשר בין מבנה הנתונים הפיסי אשר נמצא ביחידות האחסון לבין תמונת הנתונים המוצגת לתוכניות היישום או למשתמש בתקשורת מנוהל על ידי המערכת לניהול בסיס הנתונים. מבנה הנתונים במאגר מאפשר להוסיף נתונים הנדרשים ליישומים קיימים, או לשם פיתוח יישומים חדשים, מבלי שיהיה צורך לפגוע ביישומים המשתמשים בנתונים הקיימים. מערכת התוכנה לניהול בסיסי נתונים גם מאפשרת למשתמשים גישה אל הנתונים המאוחסנים בבסיס הנתונים.

מערכת התוכנה מהווה **מישק** (Interface) בין המשתמש לבין בסיס הנתונים והיא מאפשרת לו להגדיר את דרישות העיבוד מבלי שיהיה עליו לטפל בייצוג הנתונים ביחידת האחסון. התוכנה הינה אוסף של תוכניות מחשב לשם ביצוע פונקציות שונות כמו בניית בסיס הנתונים, אחזור ועדכון הנתונים בהתאם לדרישות המשתמש.



תרשים 6 – 1: מודל כללי של מערכת לניהול בסיס נתונים

ברקע של תהליכי העיבוד השוטפים המשרתים ישירות את המשתמש, כוללת מערכת DBMS מספר פונקציות נוספות. הן מאפשרות לה לתפקד בסביבה דינמית של יישומים ומשתמשים, מבטיחות את מתן השירות הנדרש ומגינות על בסיסי הנתונים מפני פגיעה בשוגג או במזיד.

—**בטיחות נתונים** (Security): כל הנתונים נמצאים במאגר אחד מרכזי ולכן נדרש פיקוח על הגישה אליו. המערכת צריכה לאפשר גישה למשתמשים המורשים בלבד ולהציג להם רק את הנתונים שהם מורשים להם לשם עיון ולשם עדכון. מערכת DBMS צריכה לאפשר לארגון לשמור על סודיות הנתונים המאוחסנים בבסיס הנתונים מפני פגיעה מכוונת או באקראי.

—**אמינות ושלימות** (Integrity): מערכת DBMS מאפשרת הגנה על אמינות ושלימות

הנתונים המאוחסנים בבסיס הנתונים. ניתן להטיל עליה לבצע חלק מבדיקות התקינות והסבירות של הנתונים לפני כתיבתם בבסיס הנתונים. קיימות בדיקות מסויימות מורכבות יותר אשר לא ניתן לבצען בצורה אוטומטית על ידי התוכנה הסטנדרטית והן מתבצעות בתוכניות היישום השונות.

— **התאוששות** (Recovery): בסיס הנתונים משרת את כל הארגון ועל כן כל פגיעה בו או הפסקת פעילותו הינם חמורים. המערכת צריכה לאפשר החזרתו לתפקוד לאחר תקלה. רוב המערכות מאפשרות את המשך התפעול גם כאשר חלק מבסיס הנתונים אינו תקין — על ידי עקיפת הקטע שנפגע. במקרים אחרים מאפשרת המערכת התאוששות מהירה של בסיס הנתונים והחזרתו לכשירות לאחר גילוי תקלה, כמו למשל הפסקת פעילות המחשב בעת העיבוד, תקלה פיסית בתקליט מגנטי, שיבוש בתוכנה וכד׳.

— **שיתוף נתונים** (Data Sharing): מערכות DBMS צריכות לאפשר למספר רב של תוכניות יישום להשתמש בו זמנית בבסיס הנתונים מבלי לפגום בסודיות, באמינות ונכונות הנתונים המאוחסנים בו. רוב המערכות מספקות כלים למניעת עדכון בו זמני של הנתונים על־ידי מספר תוכניות יישום שונות. מניעת העדכון מתבצעת על ידי סינכרוניזציה של התוכניות או על ידי חסימת קטעי מאגר. רמת שיתוף גבוהה יותר היא גישה בו־זמנית על ידי משתמשים רבים במערכות תוכנה מקוונות.

המטרות העיקריות לפיתוח והפעלה של מערכות לניהול בסיסי נתונים הן:

א. **הגדלת זמינות הנתונים:** מתן אפשרות למשתמש המנוסה ולמשתמש הפחות מנוסה לגשת בקלות ובנוחות יחסית לנתונים. רוב מערכות DBMS מספקות כלים כגון שפות שאילתות מתוחכמות **ומילון נתונים** (Data Dictionary) המכיל מידע על הנתונים המאוחסנים בבסיס הנתונים כמו למשל שמות הנתונים, משמעותם וכיצד ניתן לגשת אליהם.

ב. **יכולת לנהל את המבנה האמיתי של הנתונים:** בין הנתונים קיימים קשרים שונים, כמו למשל הקשר שבין קובץ הלקוחות לבין קובץ הזמנות, מכיון שההזמנות שייכות ללקוחות מסויימים. בגישה המקובלת לניהול נתונים קשרים אלה מנוהלים על ידי תוכניות היישום, ובגישת בסיס הנתונים אחריות ניהול הקשרים הועברה למערכת DBMS. ניהול הנתונים על־ידי התוכנה הסטנדרטית מגדיל את אמינות ונכונות הנתונים מחד, ומפשט את תוכניות היישום מאידך.

ג. **הקטנת עלויות וזמן תגובה:** מערכות DBMS מאפשרות בדרך כלל את הקטנת זמן התגובה מרגע קבלת דרישה למידע ועד לאספקת המידע למשתמש. מכיון שגישת בסיס הנתונים מבוססת על מידה רבה של אי־תלות במבנה הנתונים גם יעילות צוות התכנות עולה. במערכות DBMS מתקדמות ניתן שירות של שפת שאילתות ודיווח המאפשרת גם למשתמש שאינו מיומן להציג את דרישותיו למידע ולקבל תשובה בפרק זמן קצר יחסית. לעומת האפשרות של הקטנת זמן התגובה למשתמש המבקש מידע, יתכן שידרש זמן מחשב ארוך יותר לביצוע העיבוד בגלל הגמישות של המערכת ואי התאמה בין הדרישה לארגון הפיסי של בסיס הנתונים.

ד. **כפילות נתונים מבוקרת:** רמת כפילות נתונים גבוהה גוררת בעיות בעיבוד הנתונים ובאמינותם. גישת בסיס הנתונים מביאה רמה זו למינימום הדרוש. לעיתים, כפילות נתונים מזערית מסייעת לשיפור הביצועים של המערכת לניהול נתונים. מטרת מערכות DBMS הוא לאפשר רמת כפילות נתונים מבוקרת כדי שתהיה מוצדקת גם מבחינה כלכלית.

4. אי־תלות בנתונים

אי־תלות בנתונים (Data Independence) מוגדרת כיכולת להשתמש בנתונים מבלי להכיר את פרטי שיטת אחסונם. בסעיף זה נרחיב את הדיון במושג זה שהינו אחד המושגים החשובים ביותר בהבנת מערכות לניהול בסיסי נתונים.

לנתונים שני סוגי מבנים, המבנה הלוגי והמבנה הפיסי.

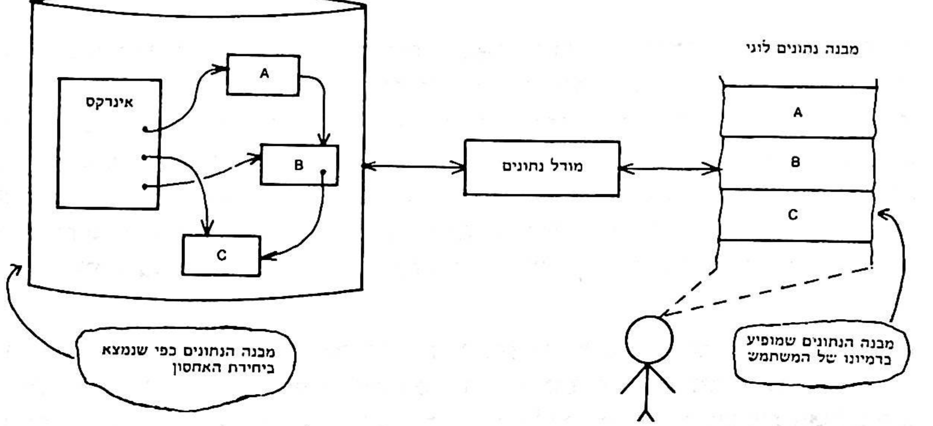
מבנה לוגי (Logical Data Structure)

המבנה הלוגי מציג מבנה נתונים כפי שהוא נראה מנקודת מבט או מתצפית המשתמש. המשתמש יכול להיות עובד מקצועי חסר ידע במחשבים אשר מנסח שאילתא באמצעות מסוף, או מתכנת הכותב תוכנית יישום. המבנה הלוגי הוא מבנה מופשט הקיים בדמיונו של המשתמש ומהווה הפשטה מסויימת של המציאות. במערכת לניהול מלאי למשל, מרכזים נתונים על רמת המלאי בכל המחסנים לכל פריט, ההזמנות לכל לקוח ועוד. כל אלה הם מושגים מופשטים אשר מיועדים לייצג את המציאות. המשתמש רואה תמיד את הנתונים במבנה מסודר כלשהו, למשל במבנה דוח על יתרות המלאי של הפריט. התוכניתן רואה בדמיונו מבנה רשומה שמכיל את הנתונים הדרושים בסדר קבוע כלשהו במסגרת הקובץ. תמונה זו יכולה להיות שונה מהמציאות מכיון שהאחסון הפיסי של הנתונים אינו בהכרח בצורה ובסדר שבה הם מוצגים במבנה הרשומה.

את המבנה הלוגי של הנתונים ניתן לתאר בצורתו המופשטת בעזרת כללים מסויימים. תיאור זה הינו **מודל הנתונים** שעליו נרחיב את הדיון בפרקים הבאים. נוכל להבחין בשני סוגים של מודלים:

א. **מודל כללי** אשר מתאר את כל הנתונים המאוחסנים במערכת. מכיון שהוא מתאר את כל הנתונים של הארגון, קיים רק מודל אחד כזה לבסיס הנתונים.

ב. **מודל חיצוני** אשר מגדיר את **התצפית של המשתמש** (User View) על הנתונים. המשתמש מתייחס בדרך כלל רק לחלק מהנתונים המופיעים במודל הכללי. לכל משתמש תצפית המתייחסת בדרך כלל רק לחלק מהנתונים המופיעים במודל הכללי. לכל משתמש תצפית שונה ולכן קיימים מספר רב של מודלים כאלה.

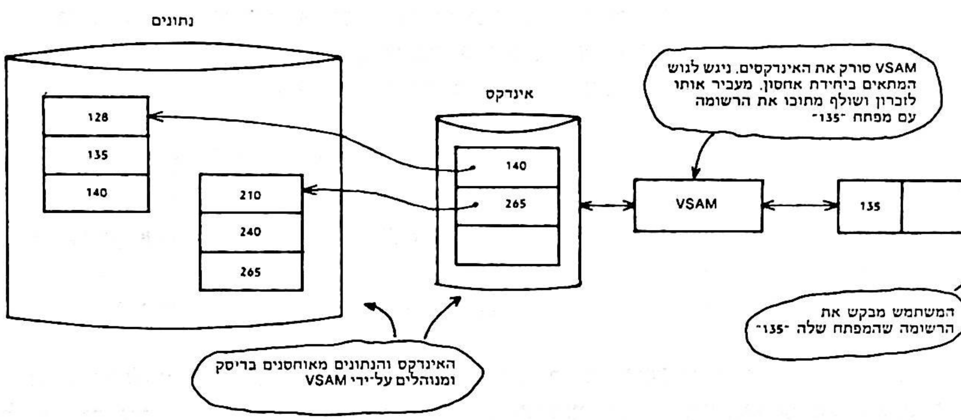


תרשים 7 – 1: תרגום מבנה פיסי למבנה לוגי

במערכות לניהול קבצים קיים רק מודל חיצוני והוא חייב להיות זהה למבנה הקובץ. רק מערכות ניהול בסיסי נתונים מאפשרות את השימוש בשני מודלים אלו.

אי־תלות פיסית מושגת על־ידי שימוש במודל אשר מבצע את התרגום בין המבנה הלוגי לבין המבנה הפיסי. תרגום זה הינו מיפוי (Mapping) של מבנה נתונים אחד במבנה נתונים אחר, כשם שהמפה הגיאוגרפית הינה מיפוי של שטח פיסי בתרשים נייר דו־ממדי.

רוב המערכות לניהול קבצים מספקות רמה זו של אי־תלות פיסית. התהליך של איתור רשומה "שקוף" בדרך כלל מבחינת המשתמש, והוא פונה אליה באמצעות השם ותוכן המפתח בלבד. צורת התרגום בין מפתח הרשומה לכתובתה ביחידת האחסון "שקוף" מבחינתו. מערכת VSAM (Virtual Storage Access Method) של יבמ למשל, מנהלת באופן אוטומטי אינדקס לאיתור מהיר של הרשומות. המערכת מתרגמת את המפתח לכתובת הפיסית על־ידי ביצוע סריקות במספר רמות של אינדקסים אשר מנוהלים על ידה.



תרשים 8 – 1: איתור רשומה לפי מפתח על ידי VSAM

קיימים שינויים במבנה הפיסי אשר מחייבים שינוי בתוכניות היישום. מעבר משיטת ארגון קובץ אינדקס סדרתי לקובץ בגישה ישירה מחייב שינוי בתוכניות מכיון שהלוגיקה שלהן מניחה שהקובץ ממויין, דבר שאינו נכון בקובץ לגישה ישירה.

מבנה פיסי (Physical Data Structure)

מבנה פיסי מציג מבנה נתונים כפי שהוא מופיע ביחידות האחסון, דיסק או סרט מגנטי למשל. מבנה זה מכיל את הייצוג הבינרי של כל הנתונים המאוחסנים במערכת ורק הוא קיים באמת במחשב. רשומת עובד למשל מאוחסנת ביחידת האחסון כרצף של סיביות (Bits) בכתובת פיסית של דיסק. כתובת זו מורכבת ממספר הדיסק, מספר הגליל, מספר המסילה והמקום היחסי בתוך המסילה. הטיפול במבנה פיסי מחייב התייחסות לנושאים כגון אורך

הרשומה, גודל הגוש, תוכן שדות, מצביעים, אינדקסים וכדומה. טיפול בסוג זה של מבנים הינו מורכב מאד, טכני מאד ומעל ליכולתו ועניינו של המשתמש. המשתמש מעדיף לחשוב במונחים של מבנים לוגיים כגון קבצים ורשומות ולא במונחים של סיביות. המבנה הפיסי יכול להיות שונה מאד מהמבנה הלוגי.

אי־תלות פיסית (Physical Data Independence)

אי־תלות פיסית מוגדרת כיכולת לבצע שינויים במבנה הפיסי מבלי שיהיה צורך לשנות את תוכניות היישום. אי־תלות פיסית נוצרת בגלל היכולת לתאר את המודל הכללי והמודל החיצוני במונחים לוגיים ללא התייחסות למבנה הפיסי. כך ניתן לתאר את קובץ העובדים במונחים של רשומות ושדות בתוך הרשומות מבלי לתאר את צורת האחסון. מכיון שהמודל אינו מתייחס להיבטים פיסיים של אחסון הנתונים נוצרת רמה מסויימת של אי־תלות במבנה הפיסי שלהם.

אפשר לבצע מספר שינויים במבנה הפיסי מבלי שיהיה צורך לשנות במקביל גם את המבנה הלוגי:

— שינוי גודל הגוש כדי שיכיל מספר רשומות גדול יותר,
— העברת הקובץ ליחידת אחסון חדשה, למשל מדיסק 3330 לדיסק 3350 של יבמ,
— ארגון מחדש של קובץ אינדקס־סדרתי כדי שכל הרשומות שגלשו תהיינה בשטח האחסון הראשי של הקובץ,
— שינוי השיטה לטיפול ברשומות גולשות,
— שינוי שיטת הארגון לשיטה דומה, למשל מ־ISAM ל־VSAM,
— הוספת אינדקסים משניים לקובץ בכדי לאפשר גישה לפי מפתחות נוספים.

אי־תלות לוגית (Logical Data Independence)

אי־תלות לוגית מוגדרת כיכולת לבצע שינויים במודל הכללי של הנתונים מבלי שיהיה צורך לשנות את תוכניות היישום. אי־התלות הלוגית קיימת בעיקר במערכות DBMS שבהן אפשרי מודל חיצוני אשר מתווך בין תוכנית היישום לבין המודל הכללי של הנתונים. המערכות לניהול קבצים אינן מאפשרות אי־תלות לוגית ורוב הבעיות שנוצרו במערכות אלה נובעות מסיבה זו.

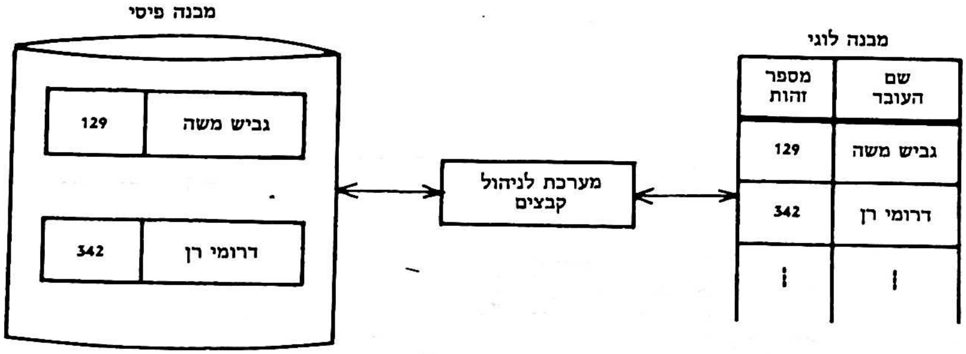
אפשר לבצע שינויים שונים במודל הנתונים הכללי ולכל אחד מהם השפעה שונה על תוכניות היישום:

— הוספת סוג רשומה חדש למודל הכללי. אין כל שינוי בתוכניות היישום הקיימות מכיון שבמודל החיצוני סוג רשומה זה לא מופיע,
— ביטול סוג רשומה קיים מהמודל הכללי. יש צורך לשנות בהתאם את כל תוכניות היישום אשר משתמשות בסוג רשומה זה,
— הוספת שדה חדש לרשומה קיימת. אין כל שינוי בתוכניות היישום הקיימות אשר המודל החיצוני שלהן אינו כולל שדה זה,
— שינוי שם השדה במודל הכללי. אין כל שינוי בתוכניות היישום הקיימות מכיון שהתרגום בין שם השדה במודל לבין שמו במודל החיצוני מתבצע אוטומטית על־ידי מערכת DBMS,
— שינוי גודל שדה כלשהו. יש צורך לשנות את כל תוכניות היישום אשר משתמשות בשדה זה. אין כל שינוי בשאר תוכניות היישום אך יתכן שיהיה צורך בהידור כי יש שינוי במיקום שאר השדות ברשומה.

– שינוי מפתח הרשומה. יש לשנות את כל תוכניות היישום מכיון שהן משתמשות במפתח לאיתור הרשומה.

כפי שניתן לראות, שינויים מסויימים במודל הכללי מחייבים שינוי בתוכניות היישום אשר משתמשות במבנה הלוגי המתאים. עם זאת קיימים מספר שינויים שאינם מחייבים שינוי בתוכניות היישום מכיון שהן משתמשות במודל החיצוני.

במערכות לניהול קבצים לא קיימים שני המודלים ולכן הוספת השדה "נסיון מקצועי" לרשומת העובד מחייבת שינוי **בכל** תוכניות היישום המשתמשות בקובץ, גם אם אין הן משתמשות בשדה זה. הצורך בשינוי נובע מהעובדה שמערכות אלה מציגות למשתמש את כל הרשומה במבנה שטוח אחד, או כפי שהיא קרוייה: **רשומה שטוחה** (Flat Record). כל רשומה פיסית מהווה גם רשומה לוגית ולכן קל יחסית לבצע את התרגום. על המשתמש להכיר את מבנה הקובץ ומבנה הרשומה, כי ללא ידיעת סדר הופעת השדות ברשומה אין הוא יודע לפענח את תוכנה.



תרשים 9 – 1: תרגום מבנה פיסי ולוגי עבור רשומות שטוחות

במערכות DBMS שבהן קיים מודל חיצוני הוספת שדה חדש לרשומה אינה מחייבת שינוי בתוכניות היישום שאינן מטפלות בשדה זה. המודל החיצוני מבודד את תוכנית היישום מהשינוי במודל הנתונים הכללי. ככל שרמת אי־התלות הלוגית גבוהה יותר, עלות ביצוע התרגום בין המבנה הלוגי והמבנה הפיסי גבוהה יותר. השאיפה היא להגיע לרמת אי־תלות כזאת המהווה פשרה בין היתרונות הנובעים ממנה לבין עלות ביצוע התרגום.

נסכם את היתרונות והחסרונות של אי־תלות בנתונים:

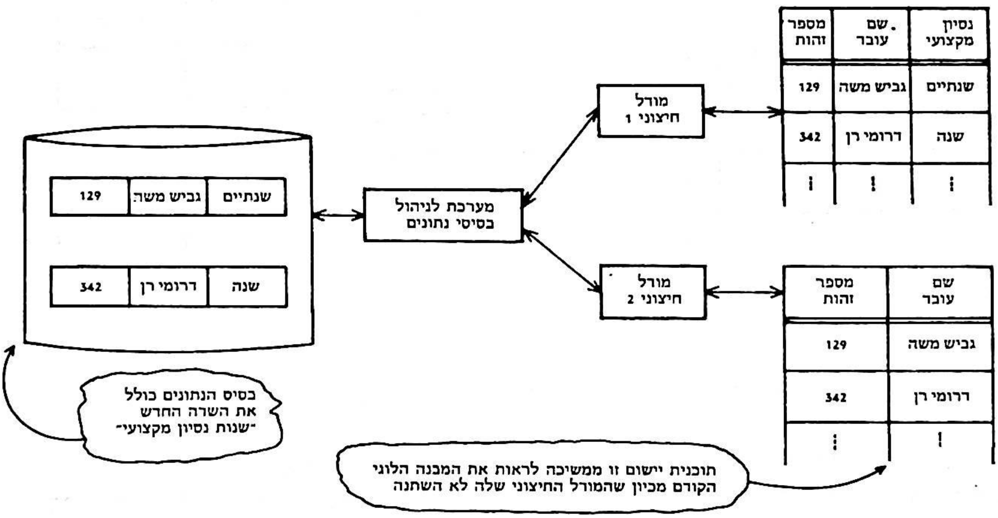
יתרונות

— נוחות שימוש מנקודת מבט המשתמש אשר אינו מטפל בבעיות הטכניות של אחסון הנתונים. פונים אל הנתונים בשמות סימליים (של רשומות, של שדות) ולא בציון כתובת ותכונות פיסיות של הרשומה.

— ניתן לבצע שינויים במבנה הפיסי של הנתונים ללא השפעה על תוכניות היישום או על איכות הפלט של התוכניות.
— ניתן לבצע שינויים מסויימים במודל הנתונים הכללי ללא צורך בביצוע שינויים בתוכניות היישום.
— אפשרות **כיוונון** (Tuning) של המערכת לניהול נתונים כדי להעלות את יעילות הביצוע של תוכניות קריטיות על חשבון תוכניות פחות קריטיות.

חסרונות

— מכיון שהמשתמש אינו מכיר את המבנה הפיסי של הנתונים הוא עלול להשתמש במערכת בצורה לא יעילה. למשל בקשה לשליפת רשומה לפי שדה שאינו שדה מפתח תגרום לסריקות סדרתיות בקובץ, וכתוצאה — לזמן עיבוד ארוך מאד.



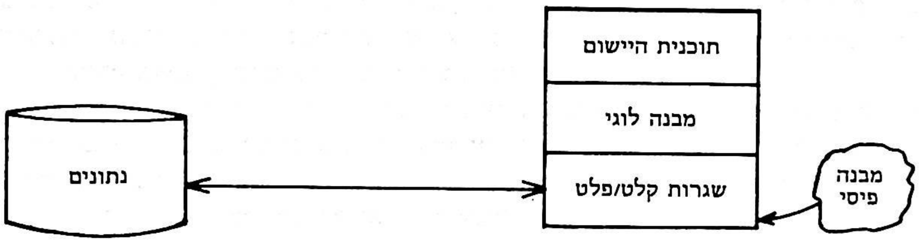
תרשים 10 – 1: תרגום מבנה פיסי ולוגי במערכות DBMS

5. ציוני דרך בהתפתחות מערכות לניהול נתונים

נסקור בקצרה את המערכות לניהול נתונים ונדגיש את המאפיינים שלהן.

5.1 מערכות בסיסיות לניהול נתונים

מערכות אלו היו קיימות עד תחילת שנות ה־60. הן דומות יותר לאוסף של הוראות קלט ופלט מאשר למערכת ניהול נתונים. כל תוכנית יישום טיפלה בקובץ אחד או יותר בהתאם לדרישות היישום וכל גישה לנתונים, הן הלוגית והן הפיסית, היתה לחלק בלתי נפרד מהתוכנית עצמה באמצעות שגרות קלט/פלט שהיו משולבות בה. השגרות נכתבו על־ידי התוכניתנים בעמל רב מכיון שטפלו ברמה הבסיסית של התכנות ובקשר שבין התוכנה לחומרה.



תרשים 11 – 1: מערכת בסיסית לניהול נתונים

תוכניות אלה היו תלויות תלות מוחלטת בצורת האחסון הפיסית של הנתונים. שיתוף נתונים בין יישומים שונים היה כמעט בלתי אפשרי. מגבלה זו גרמה לכך שקבצים שונים הכילו נתונים זהים בהתאם לצרכי היישום ולעיתים אותם הנתונים הופיעו בהם במבנים שונים (שם משפחה בקובץ אחד הכיל 40 תווים, ובשני — 35 תווים). ככל שמספר היישומים הלך וגדל, כן גדלה גם כפילות הנתונים ונפגעה אמינותם. תוכניות שונות עדכנו בנפרד את אותם הנתונים שהופיעו בקבצים שונים יגרמו לעיתים לאי התאמה. המאפיין המרכזי של מערכות אלה היה הקושי הרב לגישה לנתונים ולניהולם.

5.2 מערכות לניהול קבצים

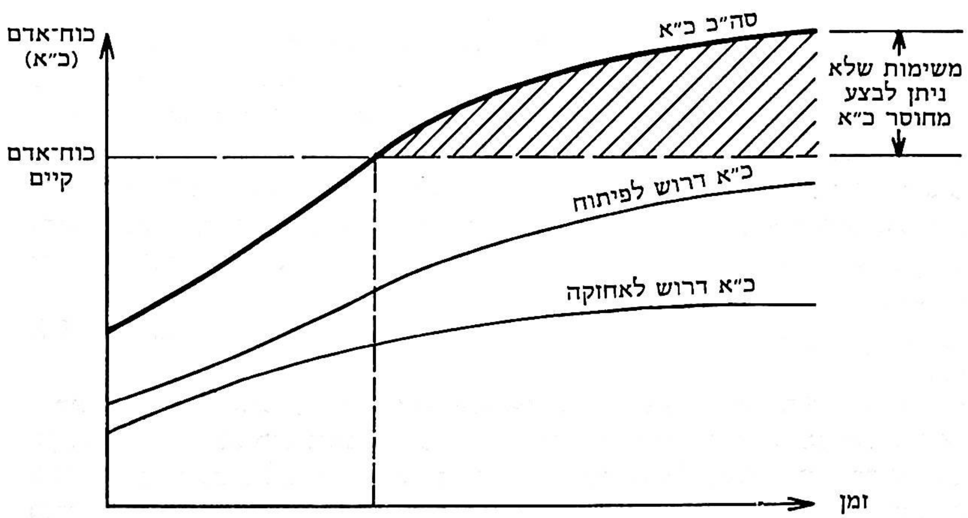
מערכות ניהול קבצים (FMS—File Management Systems) הופעלו מתחילת שנות ה־60 והן נמצאות עדיין בשימוש נרחב. הן מציעות למשתמש מגוון קטן יחסית של מבנים לוגיים אך משחררות אותו מטיפול במבנה הפיסי של הנתונים. מערכות אלו מטפלות בהצלחה בקבצים בעלי תבנית רשומה זהה או משתנה ומספקות כלים תקניים ליצירת ואחזקת הקובץ. תוכניות היישום אינן מתייחסות כלל למבנה הפיסי של הנתונים אלא בעיקר למבנה הלוגי שלהם. הן משיגות זאת באמצעות שגרות סטנדרטיות לגישה לנתונים אשר מהוות חלק מהמערכת לניהול קבצים. דוגמאות אופייניות למערכות אלה הן VSAM ו־ISAM של יבמ.



תרשים 12 – 1: מערכת לניהול קבצים

במערכות לניהול קבצים קיים קשר הדוק בין המבנה הפיסי לבין המבנה הלוגי. תוכניות היישום "מַכירות" את התכונות הפיסיות של הקובץ כגון מבנה הרשומה, מפתחות הקובץ, שיטת גישה לרשומות ועוד. הקשר הקיים בין רשומות שונות בא לידי ביטוי בתוכניות היישום ובתוכניות מערכת התוכנה אשר מאחזרות את הרשומות המבוקשות. שינויים בארגון הקובץ ובסוג יחידת האחסון חייבו שינוי מקביל בתוכניות היישום.

לדוגמא, כאשר בארגון מסויים נדרשים נתונים אודות התשלומים של הלקוחות יש לבדוק באילו קבצים נמצאים הנתונים. נניח שבמקרה זה הם נמצאים בשני קבצים שונים: קובץ הלקוחות שמכיל את פרטי הלקוחות וקובץ התשלומים שמכיל את פרטי התשלום של כל הלקוחות. הקשר בין שני הקבצים האלה יתבצע על־ידי מספר הלקוח שנמצא בשני הקבצים. על המתכנת להכיר את מבנה הרשומות והשדות בשני הקבצים ואת שיטת הגישה לכל אחד מהם כדי לכתוב את הוראות הקלט/פלט המתאימות ואת הוראות העיבוד.



תרשים 13 – 1: הקצאת משאבי כח־אדם ביחידות ענ"א

תיאור הנתונים שהתוכנית השתמשה בהם והמבנה הפנימי של כל הרשומה היו לחלק ממנה ללא אפשרות להתעלם מנתונים אפילו אינם דרושים. עם זאת שחררו מערכות אלה את התוכניתן מבעיות של הצגה פיסית של הנתונים ביחידת האחסון. שיתוף נתונים בין יישומים שונים התאפשר ע״י השימוש במערכת לניהול קבצים והושג צמצום מסויים בכפילות הנתונים. בעיות בניהול מערך הנתונים עדיין נותרו, וביניהן תיאום בין המשתמשים השונים לשם עריכת כל שינוי במבנה. הוספת נתון לקובץ גרר שינוי במספר רב של תוכניות, גם אם הנתון לא נדרש להן, והסיבה—שינוי מבנה הרשומה. הקושי בביצוע שינויים ברשומות גרם לכך שהיתה נטייה לדחות ביצוע שינויים בקבצים לאחר הפעלת היישום, וכתוצאה מכך — תכולתם לא התאימה לדרישות הצרכנים.

תרשים 13—1 מציג את המצב האופייני ברוב היחידות המשתמשות במערכות לניהול קבצים. כמות כוח האדם הדרושה לפיתוח יישומים חדשים ולאחזקת יישומים קיימים גדולה מכמות כוח האדם הקיימת ביחידה ולכן חלק גדול מעבודות הפיתוח אינו יכול להתבצע.

5.3 מערכות לניהול בסיסי נתונים

מערכות ניהול בסיסי נתונים הופיעו לראשונה בסוף שנות ה־60. היו אלו IMS של יבמ ו־IDS של הוניוול. הן הקלו במעט על המשתמשים אם כי עדיין לא הציגו רמה גבוהה של אי־תלות בנתונים. מאמץ מחקר רב הושקע על־ידי מוסדות מחקר, בתי תוכנה וארגונים שונים לקידום הידע ולהגדרת הדרישות ממערכות לניהול נתונים. בתחילת שנות ה־70 פורסמו ההצעות וההמלצות הראשונות למבנה רצוי של מערכת DBMS, מבנה שפות להגדרת נתונים ונושאים נוספים הקשורים בנושא בסיס הנתונים.

בשנת 1970 גובשה ההצעה המשותפת של שני ארגוני המשתמשים GUIDE ו־SHARE. לאחר מכן הוגשו ההמלצות של CODASYL (Conference On Data and SYstem Language), הארגון שהיה אחראי לפיתוח וקביעת תקנים של COBOL והחליט לעסוק גם בבעיות של ניהול נתונים. CODASYL הקים צוות משימה מיוחד (DBTG—Data Base Task Group) לשם הגשת הצעה למודל נתונים רצוי במערכת לניהול בסיסי נתונים. הצוות חקר מספר מערכות ניהול קבצים וניהול בסיסי נתונים שהיו קיימות ובשנת 1971 פרסם דוח שבו הגיש את המלצותיו. הדוח כלל ניסות פורמלי של מודל הנתונים הרשתי וקבע את החלוקה בין **הסכימה המתארת את המודל הכללי לבין תת־הסכימה המתארת את המודל החיצוני ואת מבנה השפה להגדרת נתונים**. מודל הנתונים של CODASYL זכה להערכה רבה מאחר והוא הציג לראשונה כיוון כללי לתכנון בסיסי נתונים וטפל במספר בעיות עקרוניות חשובות. למרות שהמודל זכה גם לביקורת רבה הוא מהווה ללא ספק ציון דרך בהתפתחות מערכות DBMS ונראה שהוא ישפיע גם על מערכות DBMS עתידיות.

בשנת 1975 הוצגה העבודה של ארגון התקינה הלאומי של ארה״ב — ANSI (American National Standards Institute). במסגרת הארגון פעלה קבוצת מחקר למערכות מחשב ומידע, צוות SPARC/X3 (Standards Planning And Requirements Commitee on Computers and Information Processing), שעסק בגיבוש מסגרת לארכיטקטורה של מערכת DBMS. עבודה זו הציגה באופן פורמלי מערכת DBMS המורכבת משלש רמות שונות. ברמה העליונה נמצאות מספר **סכימות חיצוניות** המתאימות למבנה הנתונים המבוקש על־ידי המשתמש. ברמת הביניים נמצאת **סכימה מושגית** (Conceptual) אשר מציגה את המודל הכללי של הנתונים ואינה תלוייה בתוכניות היישום המשתמשות בהם. ברמה הנמוכה

ביותר נמצאת **סכימה פנימית** המתארת את המבנה הפיסי של בסיס הנתונים. בין שלש הרמות האלה מבצעת מערכת DBMS את כל המיפויים או התרגומים הדרושים. מודל ANSI/SPARC/X3 הינו מודל מקובל כיום לארכיטקטורה של מערכת DBMS מכיון שהוא מאפשר מידה רבה של אי־תלות בנתונים, הן מבחינה הפיסית והן מהבחינה הלוגית, באמצעות שיטת המיפוי בין שלש רמות המודל. **תרשים 6 – 1** מציג את הארכיטקטורה של מערכת DBMS המבוססת על מודל זה.

המודל של ANSI היה רעיון חדשני ופריצת דרך חשובה במובן העקרוני של הבנת מבנה מערכת בסיס נתונים, אך הוא לא שימש בסיס למערכת תוכנה כלשהי. רוב המערכות המסחריות הקיימות לא מבצעות את ההפרדה לשלשת הרמות. בדרך כלל הן משלבות את המודל הפנימי עם המודל הכללי ולעיתים גם לא מספקות כלל מודל חיצוני. זו אחת הסיבות לכך שברוב המערכות הקיימות עדיין לא הושגה רמה משביעת רצון של אי־תלות בנתונים. יש להניח שבעתיד תופענה מערכות המבוססות על ארכיטקטורת ANSI/SPARC/X3 והמשתמש ישיג רמה גבוהה יותר של אי־תלות פיסית ולוגית.

המפתח להצלחת מערכות DBMS עתידיות יהיה בשני מישורים: אי־התלות בנתונים ואחזור נוח של נתונים. הן חייבות לספק מגוון כלים שיאפשרו גישה מהירה ונוחה, בעיקר לשאילתות מזדמנות, עבור משתמשים בלתי מנוסים. המערכות הקיימות מטפלות בשתי בעיות אלה ברמות שונות של הצלחה. המערכות המבוססות על מודל הנתונים הטבלאי ושפות השאילתות שלהן הינן הפיתוח האחרון בשלב זה ומכוונות להגשמת הדרישות האלו.

הבעייה העיקרית במערכות DBMS היא שהציפיות של המשתמשים והצרכנים היו גבוהות מאשר יכולת ההגשמה שלהן בפועל. ארגונים שהאמינו שמערכות אלה מהוות את התשובה לכל בעיות עיבוד הנתונים התאכזבו עד מהירה. בעיות מסויימות אמנם נפתרו אולם בעיות חדשות צצו. למרות זאת אין ספק שמערכות DBMS מהוות אבן דרך חשובה בהתפתחות המערכות לניהול נתונים בפרט ומערכות עיבוד הנתונים בכלל.

5.4 מאפיינים של מערכות לניהול נתונים

א. מערכות בסיסיות לניהול נתונים

— קבצים מאורגנים במבנה סדרתי,
— מבנה פיסי הדומה למבנה הלוגי,
— עיבוד אצווה (Batch) בלבד,
— שמירת מספר עותקים של אותו קובץ בשיטת אב־בן,
— מערכות הפעלה תומכות בפעולות קלט ופלט בסיסיות בלבד,
— שינוי במבנה הנתונים מחייב תכנות מחדש של תוכניות היישום או ביצוע שינויים גדולים בתוכניות אלו,
— מבנה הנתונים נקבע לפי צרכי יישום אחר בלבד, או בעיקר לפי צרכי יישום זה,
— כפילות רבה של נתונים.

ב. מערכות לניהול קבצים

— גישה סדרתית או ישירה לרשומות,
— קיימת אי־תלות פיסית מסויימת, ניתן לשנות יחידת אחסון למשל ללא שינוי כלשהו בתוכניות או שינוי מועט בקטעי הגדרה של יחידת האחסון,

— עיבודי אצווה ועיבודים מקוונים (On-Line),
— מערכת ההפעלה מספקת את הניהול הפיסי של יחידות האחסון,
— מבנה הנתונים נקבע לפי צרכי יישום אחד בלבד או בעיקר לפי צרכי יישום זה,
— כפילות רבה של נתונים,
— אחזור נתונים בדרך כלל לפי מפתח אחד,
— ניהול הקשרים בין הנתונים באחריות תוכניות היישום.

ג. מערכות לניהול בסיסי נתונים:

— ניתן לבנות מספר קבצים לוגיים שונים מאותו מבנה פיסי,
— מידה רבה של אי־תלות פיסית,
— אי־תלות לוגית מסויימת, קיימים שינויים כלשהם במבנה הלוגי שאינם גוררים שינויים בתוכנות היישום,
— עיבודי אצווה ובעיקר עיבודים מקוונים,
— שימוש בטכניקות ארגון קבצים מורכבות ללא ביטוי בתוכניות היישום,
— שיתוף נתונים בין מספר רב של יישומים,
— כפילות נתונים מבוקרת,
— כללים להרשאת גישה של המשתמשים בבסיס הנתונים,
— ניהול הקשרים בין הנתונים נעשה במערכת התוכנה ולא בתוכניות ביישום.

6. מנהל בסיס הנתונים

בגישה המסורתית לניהול נתונים היתה האחריות לנתונים נתונה בידי המחלקות או היישומים השונים. כל מנהל יישום שמר וניהל בהתאם לצרכיו את הנתונים אשר נדרשו לו בתהליכי העיבוד השונים. הגידול המתמיד במורכבות ובגודל היישומים הציב בעיות שליטה קשות בפני מנהלי היישומים, אך רק הופעת מערכות DBMS הביאה לשינוי בגישה המקובלת לניהול נתונים. אפשרות ריכוז הנתונים של הארגון במאגר אחד גרמה לכך שהם הוכרו כמשאב של הארגון והופקעו מבעלות מחלקתית. המורכבות הרבה של מבני הנתונים, הקשרים ביניהם והרגישות הנובעת מריכוזם למאגר משותף אחד הביאו ליצירת תפקיד חדש, **מנהל בסיס הנתונים** (DBA—Data Base Administrator). חשוב להדגיש שמנהל בסיס הנתונים **אינו** הבעלים של הנתונים שנמצאים במאגר אלא רק אחראי לניהולם התקין והיעיל ולאבטחתם. תוכן הנתונים במאגר נשאר באחריות היישומים השונים אשר כוללת הזנה, עדכון והרשאת שימוש בהם.

מנהל בסיס הנתונים מעורב בכל שלבי ההקמה, התכנון, היישום והבקרה של מערכות DBMS. מעורבותו נובעת מריכוז כל הנתונים במאגר אחד ומהצורך שמומחה מקצועי יתפעל ויפקח באופן שוטף על מערכת DBMS ויסייע לבעלי היישומים המשתמשים במאגר הנתונים. חשיבות התפקיד עולה ככל שהמערכת מורכבת יותר, ובמערכות גדולות ומורכבות ממלא את התפקיד צוות אנשי מקצוע בתחום מאגרי נתונים, כל אחד בעל תחום התמחות שונה. למנהל בסיס הנתונים תפקיד חשוב בשלבי התכנון והבחירה, בשלב עיצוב בסיס הנתונים ובעת ההפעלה השוטפת של המערכת לניהול בסיס הנתונים.

בשלב התכנון הראשוני והבחירה של מערכת DBMS משתתף מנהל בסיס הנתונים בקביעת המערכת אשר תתאים ביותר לצרכי הארגון. הוא משתתף בתהליכי התכנון לטווח ארוך של צרכי המידע בארגון ובכך יכול לשלב את צרכי הארגון עם אפשרויות היישום במערכות ניהול בסיסי נתונים לטווח הקרוב ולטווח הרחוק.

בשלב עיצוב בסיס הנתונים שותף מנהל בסיס הנתונים בתהליך הגדרת המודל הכללי של הנתונים בארגון אשר מבוסס על צרכי המידע ועל דרישות העיבוד של המשתמשים השונים. מנהל בסיס הנתונים אחראי על הקמת מודל הנתונים הלוגי אשר מתאר את הנתונים של הארגון בצורה עקבית ושלימה ומהווה בסיס לפיתוח כל היישומים העתידיים.

השלב הבא הינו הגדרת המודל הפיסי של הנתונים. מכיון שלא ניתן לפתח מבנים פיסיים שיאפשרו עיבוד יעיל בכל היישומים עליו לקבוע אילו יישומים עדיפים מבחינת דרישות העיבוד ובהתאם לכך לבנות את בסיס הנתונים. מנהל בסיס הנתונים מגדיר בתיאום עם בעלי היישומים את כללי הבטיחות במערכות ואת נוהלי הגישה לנתונים לקריאה ולעדכון.

בשלב הפעלת המערכת עורך מנהל בסיס הנתונים מעקב מתמיד כדי לשפר ולייעל וכדי להעלות את רמת האמינות והזמינות. עליו להציע שינויים שונים בנוהלי הפעלה או בתוכנה או בחומרה לאחר שהוא מעריך את התועלת הצפוייה לעומת עלות השינוי. לדוגמא, יתכן שניתן להוסיף אינדקס משני כדי ליעל את הגישה לנתונים מסויימים אבל מאידך אינדקסים אלה משפיעים על זמן העדכון וצורכים שטחי אחסון. דוגמא אחרת יכולה להיות החלטה על כפילות נתונים במטרה ליעל ולשפר את זמן הגישה. מכיון שקיימת אי־תלות פיסית המשתמש אינו מכיר את המבנים הפיסיים ולכן אינו יכול להחליט אם הוא משתמש בבסיס הנתונים בצורה יעילה. למנהל בסיס הנתונים יש את הידע על המבנה הפנימי ולכן הוא צריך לשקוד על שיפורו המתמיד. עליו לבצע שינויים שונים במודל הנתונים בהתאם לצרכים המשתנים של הארגון ולתת תמיכה לפיתוח יישומים חדשים.

תפקיד נוסף הוא פיתוח כלים לשמירת אמינות בסיס הנתונים על־ידי שימוש בחומרה ובתוכנה לגיבוי והתאוששות ולמעקב אחר גישת המשתמשים לבסיס הנתונים.

7. סיכום

מערכות המידע מבטאות יחסי גומלין בין אנשים, נתונים, כללים לעיבוד נתונים ומכונות לעיבוד נתונים. מטרות מערכת המידע הן לאחסן, לעבד ולשלוף נתונים הן לצרכים תפעוליים של הארגון והן להקטנת מרכיב אי־הודאות בתהליך קבלת החלטות בדרגי הניהול השונים.

להשגת המטרות, משתמשת מערכת המידע באנשי מקצוע, הן בנושאים יחודיים לארגון והן בנושאי מחשב, ובתוכניות יישום המשמשות לרישום לעיבוד ולאחזור נתונים בהתאם לדרוש. החלק הקריטי ביותר במערכת מידע הינו בדרך כלל החלק המטפל באחסון ובאחזור הנתונים, בעדכון ובהגנה על שלימותם.

בתחילת תקופת השימוש במחשבים לעיבוד נתונים מסחריים כל תוכנית יישום טיפלה באופן ישיר בנתונים. כתוצאה מכך נוצרה כפילות נתונים רבה והושקע מאמץ ניכר בפיתוח יישומים. שיתוף נתונים בין יישומים שונים היה קשה מאד וכמעט בלתי אפשרי. כתוצאה מכך התפתחו המערכות לניהול קבצים אשר שחררו את המשתמש מטיפול ישיר במבנה הפיסי. למרות זאת, גם הן חייבו ידע מסויים מצד המשתמש על צורת אחסון הנתונים ועל שיטות הגישה אליהם, על מבנה הרשומה וסדר השדות בה ועוד. עם זאת, הושגה התקדמות רבה בתחום אי־התלות הפיסית של תוכניות היישום אשר ביטאו במידה רבה יותר את התהליך הלוגי של הטיפול בנתונים.

מערכות DBMS התפתחו כדי לאפשר גישה קלה ונוחה יותר לנתונים. מערכות אלה משמשות כמישק בין תוכנית היישום לבין המבנה הפיסי של בסיס הנתונים. המשתמש מציין באילו נתונים הוא רוצה מבלי שיהיה עליו לפרט היכן הם נמצאים וכיצד יש לגשת אליהם. חשוב להדגיש שלמרות העובדה שמערכות DBMS אינן מערכות מידע הן מהוות כלי טכנולוגי חשוב בדרך ליישום מערכות מידע יעילות.

היישום המוצלח של מערכות DBMS מחייב תכנון מפורט תוך שימת דגש לתכנון בסיס הנתונים, פיתוח מודולרי של היישומים ותמיכה טובה של אנשי תוכנה. לא כל היישומים של מערכות DBMS היו מוצלחים ויש ארגונים שאינם מעיזים להפעיל מערכות אלה. לעומת זאת ניתן למצוא מספר רב של הצלחות בהקמת מערכות DBMS והפעלתן במשך שנים רבות.

המעבר ממערכת לניהול קבצים למערכת DBMS, כלומר לבסיס נתונים משולב מהווה שינוי משמעותי בתהליך עיבוד הנתונים בארגון. אין להתעלם גם מהמרכיב הכספי הכרוך בשינוי זה — צורך במומחים בנושא, עלות התוכנה וצריכת משאבים גדולה יותר. גורמי העלות ניתנים לזיהוי ולכימות בקלות יחסית לעומת גורמי התועלת. בין גורמי התועלת הישירים אפשר למנות את החסכון בתכנות של המערכות לניהול קבצי הנתונים ושילובם למטרות הארגון, חסכון באחזקה ובשינויים מקיפים כאשר נדרש שינוי במערכת או בחומרה ועוד. התועלת העקיפה הינה בהעלאת רמת האמינות והזמינות של המערכת, גמישות יחסית לשינויים בנתונים, ביישומים או בחומרה.

בחירת מערכת DBMS מתאימה לארגון הינה החלטה חשובה מאד כי היא קובעת את שיטת ניהול הנתונים לטווח ארוך. מערכת זו הינה "קו הייצור" של המידע בארגון וכשם ששינוי קו ייצור כרוך בקשיים כן מעבר ממערכת DBMS אחת לאחרת.

8. שאלות

1. מהו המונח המתאים להגדרת "בסיס נתונים":
   - בלתי תלוי בחומרה של המחשב ובאילוצי תוכנה.
   - מכסה את כל סוגי הנתונים עבור כל המשתמשים.
   - הנתונים המאוחסנים חייבים לייצג את המציאות.
   - יש לדאוג לא לאבד נתונים מאוחסנים ולתת להם להפוך לבלתי אמינים בעקבות שגיאת מחשב או אדם.
   - אנשים וארגונים מצפים שהנתונים אודותיהם לא ישמשו למטרות אסורות ולא יוצגו בפני אנשים בלתי מורשים.
2. מדוע רצוי שבבסיס הנתונים יאחסן כל עובדה פעם אחת בלבד?
3. תאר את ההבדל בין אי־תלות לוגית לאי־תלות פיסית.
4. האם ניתן להשתמש בגישת בסיס הנתונים רק בארגונים גדולים? מה מידת החשיבות וההגיון בגישה זו עבור ארגונים קטנים?
5. במה שונה בסיס הנתונים מקובץ המכיל רשומות הבנויות לשירות יעיל של יישום מסויים בלבד?
6. הצג אוסף של טיעונים בהם היית משתמש כדי לשכנע מנהל להשתמש בגישת בסיס הנתונים למערכות המידע של הארגון.

פרק ב

אחסון נתונים וארגון קבצים

1. מבוא
2. מושגי יסוד
3. ארגון קבצים בסיסי
4. ארגון קבצים עילי
5. שאלות

1. מבוא

הגורמים המשפיעים בצורה ניכרת על הביצועים של תוכניות היישום המופעלות בסביבת בסיס הנתונים הם שיטות אחסון הנתונים ושיטות הגישה אליהם. מערכות DBMS משתמשות במודל נתונים פנימי כלשהו המיושם באמצעות שיטות הגישה התקניות של מערכת ההפעלה. רוב המערכות מציעות למתכנן מספר שיטות גישה שונות ועליו לבחור את השיטה המתאימה לו ביותר.

מטרת התכנון של המבנה הפיסי של בסיס הנתונים היא בניית בסיס נתונים אשר יכול להשיג את הביצועים הטובים ביותר בעלות הנמוכה ביותר. ככל שבסיס הנתונים יהיה יותר קטן מבחינת נפח אחסון יהיה צורך בפחות יחידות אחסון ולכן העלות של אחזקת הנתונים תהיה נמוכה יותר. מבנה נתונים אשר מאפשר איתור מהיר של רשומות יקטין את עלות העיבוד ויעלה את שביעות רצון המשתמשים. חשוב לזכור שבסיס הנתונים משרת מספר רב של יישומים בעלי דרישות עיבוד שונות ולכן מבנה הנתונים חייב לייצג מידה מסויימת של פשרה כדי להשיג אופטימום על פני כל היישומים ולא עבור יישום אחד בלבד.

בדרך כלל הטכניקות המאפשרות גישה מהירה לרשימות מחייבות אחסון של אינדקסים או מצביעים (בכך מגדילים את נפח האחסון של בסיס הנתונים). שיפור זמן הגישה לרשומות יכול לבוא לעיתים על חשבון זמן העדכון בגלל הצורך לנהל אינדקסים נוספים. המתכנן חייב לדעת את היתרונות והחסרונות של שיטות הגישה השונות ולבחור את השיטה המשיגה את הביצועים הטובים ביותר, בעלות הנמוכה ביותר עבור כל היישומים המופעלים והמתוכננים במערכת.

בפרק זה נסקור את השיטות השונות לארגון קבצים ואת שיטות הגישה לנתונים המיושמות ברוב מערכות DBMS. השיטות נסקרות בקצרה רבה והקורא המעוניין מופנה לספרות הענפה בנושא ארגון קבצים ומבנה נתונים, ואל הספר "ארגון נתונים וניהול קבצים" מאת המחבר (הוצאת "הוד־עמי", 1982).

2. מושגי יסוד

בסעיף זה נציג את מושגי היסוד הקשורים באחסון ובניהול נתונים. מקובל לחלק את המושגים לשתי קבוצות:

—מושגים הקשורים **ליחידות האחסון**, כלומר **לחומרה** המשמשת לאחסון הנתונים.

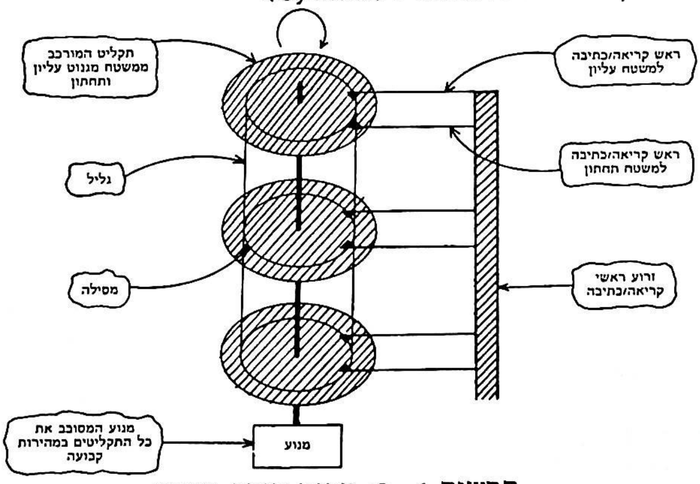
—מושגים הקשורים **למבנה הנתונים**. בקבוצה זו מקובל להבחין בין מבנה הנתונים כפי שהם מאוחסנים ביחידות האחסון—**המבנה הפיסי**, לבין מבנה הנתונים כפי שהוא מוצג למשתמש—**המבנה הלוגי**.

2.1 מושגי יסוד ביחידות אחסון נתונים

הזכרון הפנימי של המחשב הינו קטן יחסית בהשוואה לכמויות הנתונים שיש לאחסן ויקר מאד לאחסון נתונים לאורך זמן. לכן פותחו יחידות אחסון חיצוניות המשמשות לאחסון כמויות גדולות מאד של נתונים בעלות נמוכה יחסית. יחידות אחסון אלו כוללות מספר סוגי ציוד: סרטים מגנטיים, דיסקים מגנטיים, תופים מגנטיים, קסטות, תקליטונים וכדומה. בהמשך נדון **ביחידות הדיסקים** (Magnetic Disk Units) מכיון שבסיסי הנתונים מאוחסנים ביחידות אלה אשר מאפשרים גישה ישירה לנתונים ובכך מספקים זמן גישה קצר.

יחידת אחסון בגישה ישירה (DASD—Direct Access Storage Device) היא יחידה שבה ניתן לגשת בצורה ישירה לנתונים הדרושים באמצעות כתובת, בלי צורך לסרוק או לקרוא את כל הנתונים בקובץ. דיסקים ותקליטונים הם יחידות אחסון בגישה ישירה. סרט מגנטי למשל, אינו מאפשר גישה ישירה לנתונים כדי לאתר נתונים בסרט מגנטי יש לקרוא את כל הנתונים עד שמגיעים לנתונים המבוקשים.

מערכת הדיסק המגנטי מורכבת ממספר **תקליטים** בעלי משטחי מיגנוט (הדומים לתקליטי מוסיקה) הסובבים על ציר משותף. כל משטח מגנוט מחולק **למסילות** (Tracks) שבהן ניתן ל"רשום" את הנתונים. לכל משטח מגנוט יש ראש קריאה/כתיבה המאפשר את כתיבת הנתונים במסילות ואת קריאתן לאחר מכן. ראשי הקריאה/כתיבה של כל משטחי המגנוט קשורים לזרוע אחת. אוסף כל המסילות בכל משטחי המגנוט שניתן לקרוא או לכתוב מבלי להזיז את ראשי הקריאה/כתיבה הוא **גליל** (Cylinder).



תרשים 1 – 2: מבנה דיסק מגנטי

זמן הגישה לנתונים בדיסק מורכב משתי יחידות: **זמן הזזת הזרוע עד להבאת** ראשי הקריאה/כתיבה אל הגליל שמאחת ממסילותיו יש לקרוא **והזמן הדרוש לתנועת הסיבוב** של המערכת עד שהנתונים עוברים מתחת לראש הקריאה/כתיבה. זמן הזזת הזרועות הינו הגדול מביניהם ולו השפעה רבה על זמן הגישה הנתונים בדיסק. הזמן הממוצע המקובל ביחידות דיסקים חדשות הוא כ־20 אלפיות שניה. כדי לקצר את זמן הגישה פותחו מערכות דיסקים שבהן יש לכל מסילה ראש קריאה/כתיבה נפרד ולכן אין צורך בהזזת הזרועות. בדיסקים אלה זמן הגישה הינו מהיר יותר אבל היחידות יקרות יותר. בכמה ממערכות הדיסקים המוצעות בשוק ראש ראשי קבועים רק לחלק מהגלילים. בתחום ה"כיסוי" שלהם משתמשים לאחסנת קטעים ממערכות ההפעלה, שטחי עבודה של מערכות תקשורת ובמערכות ניהול נתונים מאחסנים בשטח זה את האינדקסים.

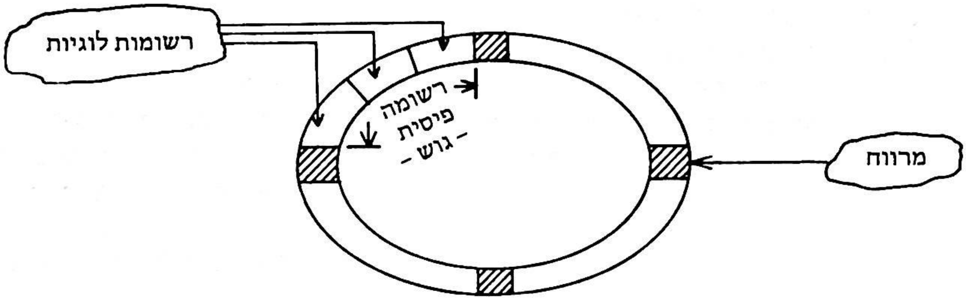
2.2 מושגי יסוד במבנה נתונים

הנתונים נרשמים ומאוחסנים ביחידות האחסון החיצוניות לפי התכונות הפיסיות שלהן (בדיסק למשל אלו הם בגליל, המסילה והגוש) ולכן זהו **המבנה הפיסי של הנתונים**. המבנה הפיסי מוכתב, אם־כך, על ידי תכונות אמצעי האחסון ולא על ידי דרישות לוגיות של תוכנת היישום של המשתמש. כדי לסייע למשתמש לטפל במבנה הנתונים פותחה **תוכנה לניהול נתונים** אשר מאפשרת לו להציג מבנים מופשטים או לוגיים. מבנים אלו שונים בדרך כלל מהמבנה הפיסי המוצג ביחידת האחסון. לכן חייבת התוכנה לבצע את ההמרה בין המבנה הפיסי של הנתונים לבין המבנה הלוגי שלהם אשר מוכר למשתמש. המבנים הלוגיים מתארים עובדות, עצמים ואירועים שאודותם רוצים לאסוף מידע. אלו הן **הישויות** (Entities) של מערכת המידע.

—מבנה לוגי:
יחידת המידע הקטנה ביותר המעניינת את מתכנן בסיס הנתונים היא **שדה** (Field). לכל תכונה של ישות המיוצגת בבסיס הנתונים יש שדה מקביל אשר מאחסן את הערך של התכונה. השדה יכול להיות בעל אורך כלשהו ולהכיל נתונים נומריים או אלפא נומריים. את השדות המתארים יישות לוגית אחת נהוג לקבץ למבנה הנקרא **רשומה** (Record) לדוגמא, כל השדות המתארים את העובד במפעל יקובצו לרשומת העובד. השדה או השדות המזהים בצורה חד־משמעית את הרשומה יקראו **שדות מפתח** (Key Fields). אוסף כל הרשומות המתארות ישות לוגית אחת יקרא **קובץ** (File). לדוגמא, כל רשומות העובד מרכיבות את קובץ העובדים.

—מבנה פיסי:
יחידת המידע הבסיסית ביחידת האחסון היא **הסיבית** (סיפרה בינרית — Bit). **הבית** (Byte) הינו אוסף של סיביות שניתן לפנות אליהן ככתובת והוא מכיל 8 סיביות. המחשב אינו קורא או כותב ליחידת האחסון רשומות לוגיות אלא רשומות פיסיות. **רשומה פיסית** או **גוש** (Block) הינם אוסף של אחת או יותר רשומות לוגיות אשר נכתבות או נקראות בפעולות קלט/פלט אחת. בין הגושים ביחידת האחסון נרשם **מרווח** (Gap) המאפשר ליחידת הבקרה של יחידת האחסון לבצע פונקציות בקרה שונות בין קריאת או כתיבת גוש אחד למשנהו.

המשתמש פונה אל הרשומות הלוגיות בציון תוכן המפתח של שדה המפתח. על התוכנה לניהול נתונים להפוך מפתח זה לכתובת פיסית של הרשומה על הדיסק.



תרשים 2 – 2: רשומות לוגיות ופיסיות בדיסק

—ארגון הקובץ:
הקובץ הינו אוסף של רשומות השייכות לנושא אחד אשר מאוחסנות ביחידת אחסון חיצונית. האחסון החיצוני נובע מן הצורך באחסון לאורך זמן ולשימוש חוזר. קבצים קטנים (טבלאות) נקראים בשלימותם לזכרון בעת העיבוד, אך קבצי הנתונים נקראים בחלקים כפי הצורך. ארגון הקובץ קובע את אפשרויות הגישה והעיבוד של הנתונים. בסרט מגנטי למשל, ארגון הקובץ מוגבל למדי. הרשומות נכתבות לפי הסדר שהן מגיעות ולכן לקריאת רשומה יש לקרוא את הרשומות שלפני הרשומה המבוקשת. הוספת רשומה חדשה אפשרית רק על ידי העתקת כל הקובץ לקובץ חדש וביצוע ההוספה תוך כדי ההעתקה אל הקובץ החדש.

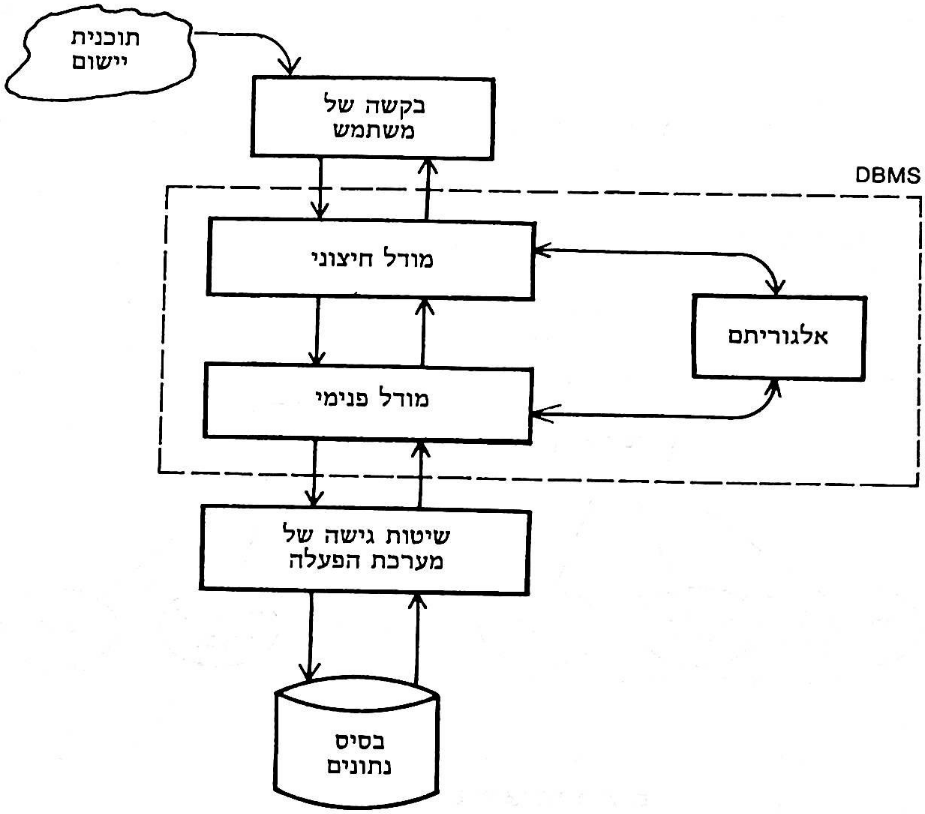
דיסקים מגנטיים מאפשרים גישה מהירה אל הרשומות על סמך ידיעת כתובתן הפיסית. רשומה זו אפשר לקרוא או לעדכן "במקומה" על ידי כתיבת רשומה חדשה באותו מקום פיסי בדיסק.

יצרני המחשבים מספקים, כחלק ממערכת ההפעלה של המחשב, שגרות מיוחדות המאפשרות בניית קבצים, גישה לרשומות בקובץ ותחזוקת הקובץ. לשגרות אלה מקובל לקרוא **שיטות גישה** (Access Method) או שגרות **לארגון וניהול רשומות** (File/Record Manager). שגרות אלו אינן מטפלות במבנה הפנימי של הרשימה, אלא רואות אותה כיחידת נתונים שלימה. פענוח המבנה הפנימי של הרשומה נעשה על ידי תוכנית המשתמש. השגרות מאפשרות למשתמש לארגן את הקובץ לפי מספר שיטות המאופיינות בעיקר לפי מהירות הגישה לרשומות והסדר לפיו ממויינות הרשומות בקובץ. בהמשך יפורטו מספר שיטות נפוצות לארגון קבצים.

2.3 גישה לרשומות במערכת DBMS

תוכנית היישום מציגה בקשה כלשהי לגישה לבסיס הנתונים, כלומר — לרשומה מסויימת, ומעבירה אותה למערכת DBMS. המערכת משתמשת במודל הנתונים החיצוני, באלגוריתם ההמרה ובמודל הנתונים הפנימי כדי לדעת אילו נתונים דרושים, היכן הם מאוחסנים ואיזו

שיטת גישה יש להפעיל כדי לקרוא אותם. לאחר פענוח הבקשה פונה מערכת DBMS אל שיטת הגישה של מערכת ההפעלה (ISAM, VSAM, BDAM וכד') לשם ביצוע הקריאה או הכתיבה בפועל. הגישה לנתונים ביחידות האחסון נעשית באמצעות שגרות קלט ופלט ובסופה של הפעולה מועבר גוש אחד או יותר של נתונים אל מאגרי קלט/פלט שבזכרון במחשב (בפעולת כתיבה יועבר גוש הנתונים מהזכרון אל יחידת האחסון). מתוך מאגרים אלה מעבירה מערכת DBMS את הנתונים המבוקשים לתוכנית היישום.



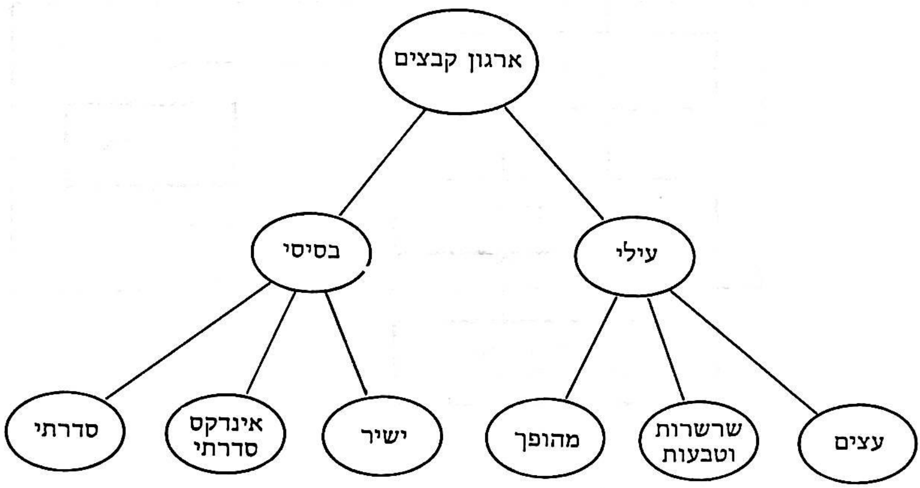
תרשים 3 – 2: גישה לבסיס הנתונים במערכת DBMS

בהמשך נסקור את שיטות הגישה המקובלות ברוב מערכות ההפעלה, אשר משמשות כבסיס למודלים הפנימיים המתוחכמים של מערכות DBMS. מקובל לחלק את שיטות הגישה לשתי קבוצות עיקריות: **שיטות גישה בסיסיות ושיטות גישה עיליות.** שיטות הגישה הבסיסיות מיושמות היום ברוב שפות התכנות העיליות (כגון PL/1, COBOL) והן כוללות שיטות הארגון הסדרתי, אינדקס סדרתי וגישה ישירה. שיטות אלה מציעות אפשרויות מוגבלות יחסית לשליפת רשומות על פי מפתח אחד אשר נקבע בעת בניית הקובץ ומכיל שדה אחד או רצף של שדות. שיטות אלה אינן עונות על בקשות שליפה מורכבות יותר, על פי שני שדות או יותר. לדוגמא, בקשה מורכבת לשליפה:

"הצג את כל העובדים אשר מועסקים במחלקה 536, גילם מעל 25 אשר עברו השתלמות בהכרת המפעל."

אם המפתח של קובץ העובדים הוא מספר תעודת זהות יש לבצע סריקה של כל הקובץ כדי לענות על שאילתא זו. כלומר, לשם פתרון בעייה כפי שמוצגת בשאילתא זו יש לאפשר גישה אל נתונים בחתכים שונים.

הצורך באחזור נתונים יעיל לפי מספר חתכים או קשרים בין הרשומות הביא לפיתוח **שיטות ארגון קבצים עיליות** (High-Level File Organization Methods). מטרתן לאפשר גישה לנתונים על פי מספר מפתחות ועל פי קשרים בין רשומות. שיטות ארגון קבצים אלו מיושמות במערכות DBMS.



תרשים 4 – 2: שיטות ארגון קבצים

3. ארגון קבצים בסיסי

קיימות שלש שיטות בסיסיות לארגון נתונים בקבצים:

— ארגון סדרתי,

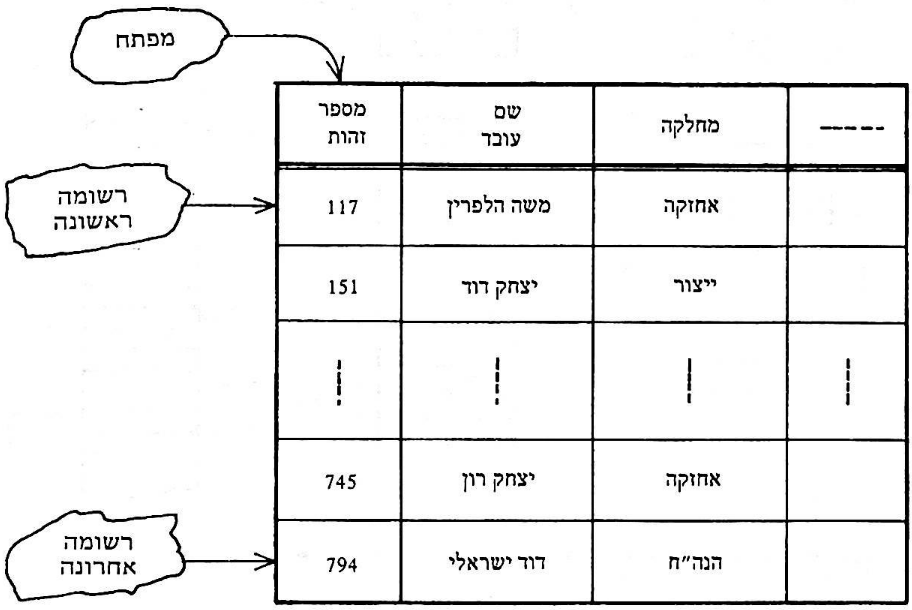
— ארגון אינדקס סדרתי,

— ארגון לגישה ישירה.

בהמשך נסקור בקצרה כל אחת משיטות אלו.

3.1 ארגון סדרתי (Sequential Organization)

השיטה הפשוטה ביותר לאחסון רשומות היא שיטת הארגון הסדרתי שבו הרשומות נכתבות אחת לאחר השניה בקירבה פיסית. סדר כתיבת הרשומות הוא שקובע את סדר קריאתן. על מנת לקרוא את הרשומות לפי סדר לוגי של מפתח יש לכתוב אותן מלכתחילה לפי הסדר הזה או למיין לפני העיבוד. נניח שקראנו רשומה מסויימת מתוך הקובץ ועכשיו יש לגשת לרשומה אחרת עם מפתח יותר גבוה. במקרה הטוב ביותר הרשומה המבוקשת היא הרשומה הבאה ובמקרה הגרוע ביותר הרשומה המבוקשת נמצאת בסוף הקובץ. מאחר ובשיטה הסדרתית כדי לגשת לרשומה n בקובץ יש לקרוא את כל n-1 הרשומות המופיעות לפני הרשומה המבוקשת, נקבל שבממוצע יש לקרוא כמחצית מהרשומות בקובץ. הגישה הסדרתית אינה מאפשרת גישה ישירה לרשומות לפי מפתח ולכן היא משמשת במערכות DBMS בעיקר למטרות גיבוי ושחזור ולא לאחסון נתונים תפעוליים.

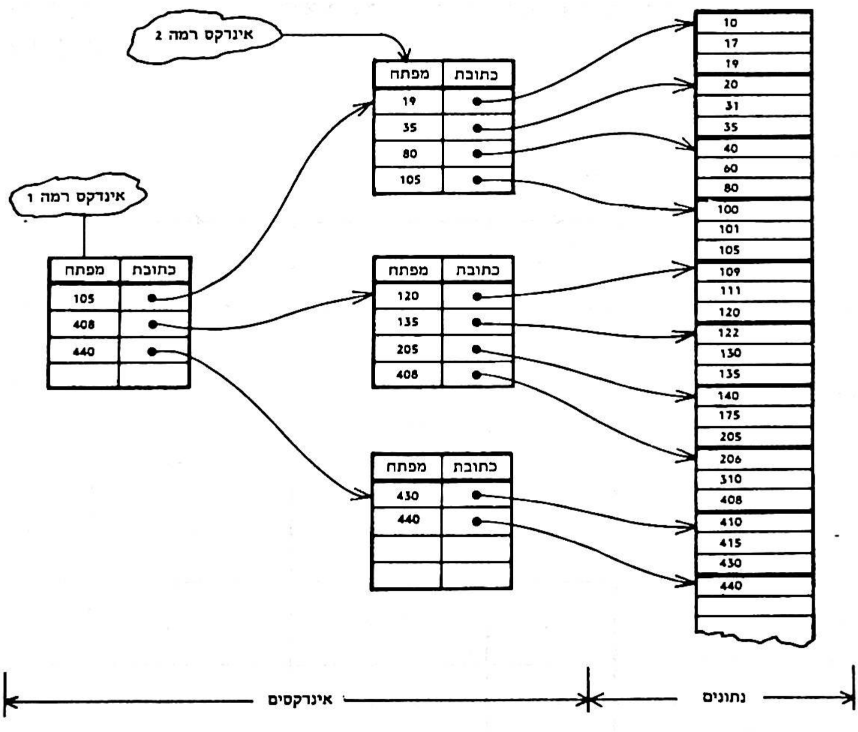


תרשים 5 – 2: קובץ סדרתי

שיטת ארגון סדרתי אפשרית בכל יחידת האחסון. שיטות הארגון האחרות אשר מאפשרות גישה ישירה לנתונים אפשריות ביחידות לגישה ישירה בלבד.

3.2 ארגון אינדקס סדרתי (Indexed Sequential Organizition)

שיטת ארגון אינדקס סדרתי משלבת את הארגון הסדרתי של הקובץ עם האפשרות לגישה מהירה לרשומה לפי המפתח שלה. בנוסף לקובץ הסדרתי בונים טבלה נפרדת, **אינדקס**, אשר מבצעת המרה בין ערך המפתח של הרשומה לבין כתובתה ביחידת האחסון. טבלת האינדקס ממויינת לפי **המפתח הראשי** (Primary Key) של הרשומה. מאחר ורשומות הנתונים ממויינות לפי אותו מפתח ומקובצות בתוך גושים אין צורך להכין כניסה אחת באינדקס עבור כל רשומת נתונים. קובעים באינדקס כניסה אחת עם המפתח הנמוך ביותר או הגבוה ביותר בגוש, ועל ידי כך אפשר להקטין את נפחו בצורה משמעותית. מאחר והאינדקס עצמו ממויין, ניתן לבנות רמות נוספות של אינדקס אשר מצביעות על קטעים הולכים וקטנים בתוכו כפי שמוצג בתרשים 6—2.



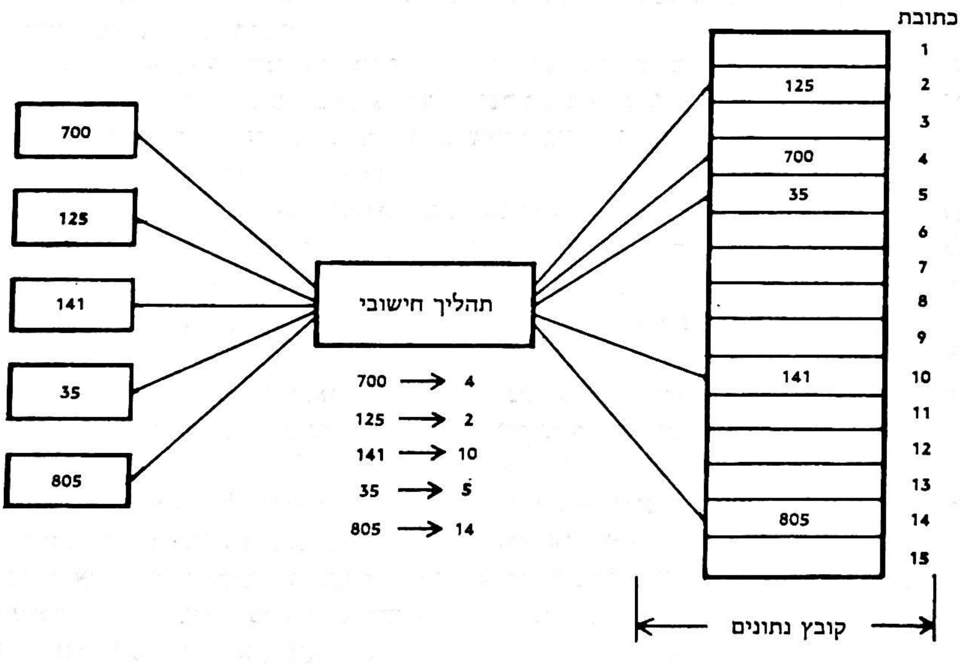
תרשים 6 - 2: מבנה קובץ אינדקס סדרתי

אחת הבעיות בארגון אינדקס סדרתי היא הוספת רשומות חדשות ושמירה על סדר מיון הרשומות. קיימות שתי שיטות לטיפול בהוספת רשומות חדשות לקובץ אינדקס סדרתי:

א. מוסיפים את הרשומה החדשה למקומה המתאים בגוש הנתונים בהתאם לסדר המיון. מאחר וגוש הנתונים מלא ברשומות, הרשומה האחרונה בגוש גולשת לגוש מיוחד הנקרא **גוש גלישה** (Overflow Block). גוש הנתונים המקורי **משורשר** באמצעות **מצביע** אל גוש הגלישה המתאים. בזמן חיפוש רשומה מתבצע חיפוש גם בגוש הגלישה הקשור לגוש הנתונים המקורי. שיטה זו מיושמת במערכת ISAM (Indexed Sequential Access Method) של יבמ.

ב. בזמן הטעינה הראשונית של הקובץ לא ממלאים את גוש הנתונים ברשומות אלא משאירים אחוז מסויים משטח הגוש ריק. כל עוד יש מקום בגוש הנתונים מוסיפים רשומות חדשות לגוש תוך שמירה על סדר המיון. אם יש להוסיף רשומה לגוש מלא מעבירים מחצית מהרשומות לגוש נתונים ריק חדש, מוסיפים את הרשומה לגוש המתאים ומעדכנים את האינדקסים בהתאם. שיטה זו מיושמת במערכת VSAM (Virtual Storage Access Method) של יבמ.

בשיטת ארגון אינדקס סדרתי ככל שהקובץ גדול יותר תהיינה יותר רמות אינדקס כדי להשיג זמן גישה טוב יותר. בשיטת ISAM אם רמת הפעילות של הקובץ היא גבוהה יהיה צורך לבצע ארגון מחדש כדי להכניס את כל הרשומות הגולשות לתוך גושי הנתונים ולחסוך זמן חיפוש בשטחי גלישה. בשיטת VSAM אין צורך לבצע ארגון מחדש מכיון ששיטת העיבוד השוטפת כוללת את השירות של ארגון מחדש.



תרשים 7 – 2: פיזור רשומות בשיטה לגישה ישירה

3.3 ארגון לגישה ישירה (Direct Access Method)

בשיטת הארגון לגישה ישירה אין צורך לבצע חיפוש באינדקסים לשם גישה לרשומה המבוקשת. התוכנה לניהול נתונים מבצעת באמצעות חישוב או טבלה את ההמרה של המפתח של הרשומה המבוקשת לכתובת של הרשומה ביחידת האחסון. לתהליך החישובי מקובל לקרוא **רנדומיזציה** (Randomizing or Hashing).

קיימות מספר רב של שיטות מתמטיות המקבלות כקלט את המפתח של הרשומה ומייצרות את הכתובת הפיסית שלה. הן מנסות למפות טווח רחב של מפתחות לתוך טווח קטן יותר של כתובות פיסיות, ולכן הן מפיקות במקרים מסויימים כתובות זהות עבור ערכים שונים של המפתח. בכל כתובת פיסית ניתן לאחסן רק רשומה אחת ולכן יש לבנות מנגנון לטיפול **בכתובות זהות** (Synonyms). אחת האפשרויות היא לאחסן רשומות כפולות אלו בגושי גלישה מיוחדים ולשרשר אותם אל הכתובת הפיסית המקורית שבה היו צריכות להיות מאוחסנות.

4. ארגון קבצים עילי

ארגון קבצים עילי הינו בדרך כלל שילוב של ארגון קבצים בסיסי כמו אינדקסים או גישה ישירה, עם שיטת מצביעים.

המצביע (Pointer) הינו שדה נוסף בתוך הרשומה אשר מוֹרֶה על רשומה אחרת במרחב הכתובות של מבנה הנתונים. המצביע קושר מספר רשומות נפרדות למבנה לוגי אחד, ולכן הוא מאפשר **הפרדה** בין המיקום הפיסי של הרשומות ביחידות האחסון לבין המבנה הלוגי של הנתונים. שדה המצביע יכול להכיל:

א. מפתח לוגי של הרשומה הבאה. מקובל לכנות מצביע זה **מצביע סימבולי.** לדוגמא אם נשמור ברשומה העובד שדה המכיל את מספר תעודת הזהות של המנהל נקבל מצביע סימבולי בין שתי רשומות. את המפתח הלוגי יש לתרגם לכתובת פיסית של הרשומה לפי אחת מהשיטות הבסיסיות, באמצעות אינדקס או לפי תהליך חישובי כלשהו.

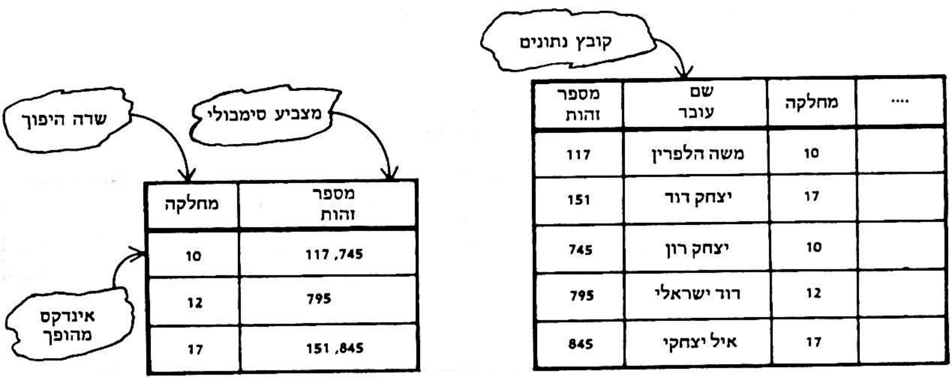
ב. כתובת פיסית של הרשומה הבאה. שדה המצביע מכיל את הכתובת הפיסית (דיסק, גליל, מסילה, גוש) של הרשומה הבאה והוא מאפשר בכך גישה מהירה מאד לרשומה המבוקשת. שינוי כתובתה של הרשומה מחייב עדכון מקביל של כל המצביעים אליה, וזו אחת מחולשותיה של השיטה.

ג. כתובת יחסית של הרשומה הבאה. ניתן למספר באופן רציף את כל הרשומות מתחילת הקובץ ולהציג בשדה המצביע את המספר הסידורי של הרשומה הבאה. ההמרה של המספר הסידורי אל הכתובת הפיסית מבוצעת על ידי מערכת ההפעלה. יתרונה של השיטה בכך שאינה מחייבת שינוי תוכן המצביעים כאשר חל שינוי בכתובת הפיסית של הרשומה. שיטה זו מקובלת מאד במערכות DBMS.

4.1 ארגון מהופך (Inverted)

הארגון המהופך מאפשר גישה לרשומה לפי מספר מפתחות (שדות) שונים לעומת השיטות שהוצגו עד כה אשר משתמשות במפתח **גישה ראשי אחד** (Primary Key). בשיטת הארגון המהופך בונים טבלה או אינדקס נפרד עבור כל אחד **משדות המפתח המשניים** (Secondary Keys). הכניסה בטבלה מכילה את ערך המפתח המשני כמפתח מיון ואת המפתח הראשי של אותה רשומה כנתון המצביע על רשומות הקובץ. בכך "מתהפך" הסדר של השימוש במפתחות.

תרשים 8—2 מציג **קובץ מהופך** עבור השדה "מחלקה" בקובץ "כוח אדם".



מספר זהות	שם עובד	מחלקה	....
117	משה הלפרין	10	
151	יצחק דוד	17	
745	יצחק רון	10	
795	דוד ישראלי	12	
845	איל יצחקי	17	

מספר זהות	מחלקה
117 ,745	10
795	12
151 ,845	17

תרשים 8 – 2: אינדקס מהופך

הגישה לאינדקס המהופך יכולה להתבצע באמצעות רמות אינדקס אחדות בדומה לשיטה המיושמת בשיטת ארגון אינדקס סדרתי. אפשר לבצע את הגישה גם באמצעות תהליך חישובי לקבלת הכתובת הפיסית של הכניסה המתאימה באינדקס. יעילות הגישה לרשומות בארגון מהופך תלויה ביעילות החיפוש באינדקס המהופך.

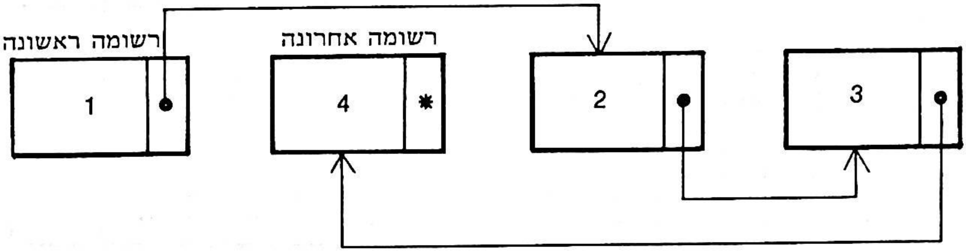
ניתן לראות שבשיטה זו רשומות הנתונים אינן מכילות כל מצביעים שהם והן מאורגנות לפי אחת משיטות הארגון הבסיסיות או שיטות הארגון העיליות. האינדקס המהופך מכיל את הערך של שדה המיון המשני (שדה ההיפוך) ואחריו שרשרת של מצביעים (סימבוליים או כתובות) לרשומות המכילות ערך זה בשדה ההיפוך.

רמת ההיפוך נקבעת על ידי היחס בין מספר שדות ההיפוך ברשומה לבין כלל השדות ברשומה. רמת היפוך "אפס" היא בפועל ארגון קובץ בסיסי לפי מפתח ראשי בלבד. רמת היפוך 1 מציינת שכל השדות ברשומה הם שדות היפוך ולכן ניתן לשלוף את הרשומה לפי כל אחד מהשדות. כלומר רמת ההיפוך מוגדרת בין 0 ל-1. קיימות מערכות DBMS המאפשרות למשתמש לקבוע את רמת ההיפוך בעוד שמערכות אחרות מאפשרות רמת היפוך 1 בלבד.

אחד היתרונות של הארגון המהופך הוא האפשרות לענות על מגוון רחב של שאילתות המבוססות על חתכים בין שדות ההיפוך. חשוב לזכור שניהול האינדקסים המהופכים מטיל מעמסה כבדה על המערכת בזמן עדכון הקובץ. אם רמת ההיפוך היא גבוהה גם נפח האינדקסים המהופכים יכול להיות גדול ויש להתחשב בו בתכנון נפח הקובץ.

4.2 שרשרות וטבעות

שרשרת (List, Chain) הינה רצף של רשומות הקשורות ביניהן באמצעות מצביעים. הרצף הסדרתי הלוגי של השרשרת אינו תלוי בכתובות הפיסיות של הרשומות.

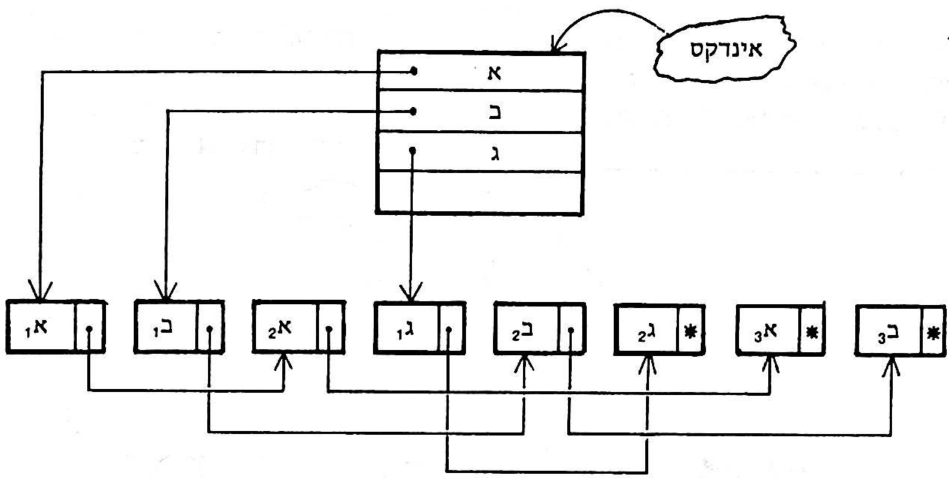


תרשים 9 – 2: מבנה שרשרת פשוטה

תרשים 9—2 מציג **שרשרת פשוטה** שבה הרצף הלוגי של הרשומות שונה מהרצף הפיסי שלהן. לדוגמא, ניתן לשרשר את כל הרשומות בקובץ העובדים לפי סדר עולה של גיל העובד בנוסף לארגון הבסיסי (אינדקס או ישיר) של הקובץ. מערכת המאפשרת שרשור רשומות צריכה לאפשר גישה **לרשומה הראשונה בשרשרת** (List Header) ולזהות את **הרשומה האחרונה בשרשרת.**

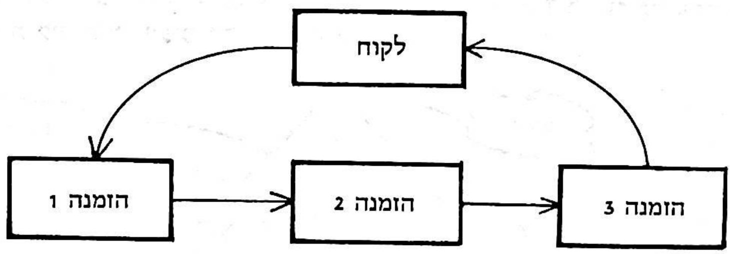
שרשרת דו־כיוונית (Two-way List) פשוטה הינה שרשרת שבה כל רשומה מצביעה על הרשומה הבאה ועל הרשומה הקודמת בשרשרת. שיטה זו מאפשרת איתור פשוט של הרשומות בשרשרת ומפשטת את תהליך העדכון ואת הוספת הרשומות ברצף.

שרשרת מורכבת (Multi-list) הינה אוסף של שרשרות פשוטות אשר מהוות יחידה לוגית אחת. לדוגמא, את רשומות העובדים במפעל נקשור במספר שרשרות פשוטות נפרדות לכל מחלקה ומחלקה. כל שרשרת מנוהלת באופן נפרד באמצעות אינדקס שבו נמצאים המצביעים אל תחילת כל אחת מהן.



תרשים 10 – 2: שרשרת מורכבת

טבעת (Ring) הינה שרשרת פשוטה שבה **הרשומה האחרונה** מצביעה חזרה אל **הרשומה הראשונה**. מבנה זה שימושי מאד במערכות DBMS. לדוגמא, מבנה טבעת הקושר רשומת לקוח לכל רשומות ההזמנה שלו. אפשר להרחיב שיטת ארגון זו על ידי מצביעים מכל רשומת טבעת אל רשומת הכותרת, ובדוגמא — מכל רשומת הזמנה אל רשומת לקוח.



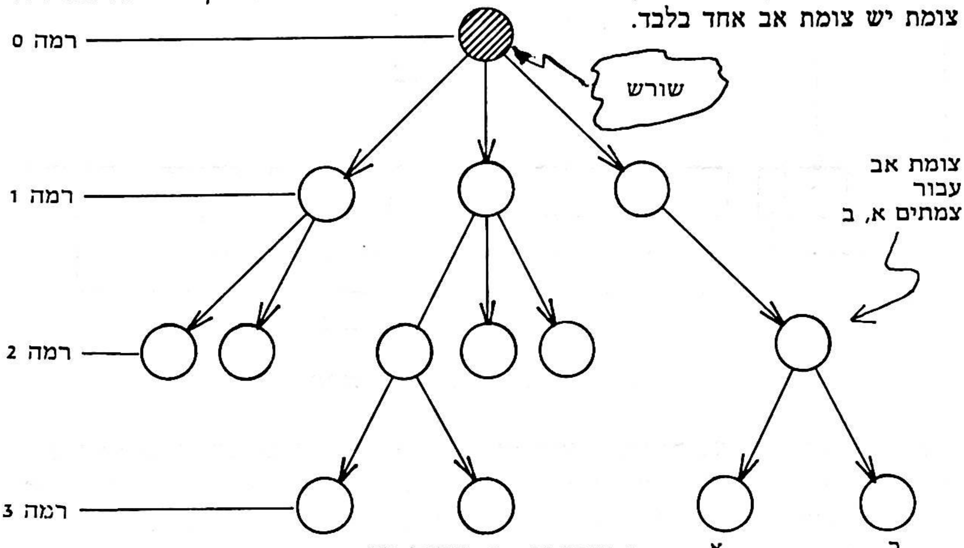
תרשים 11 – 2: טבעת פשוטה

טבעת דו־כיוונית (Two-way Ring) הינה טבעת פשוטה שבה כל רשומה מצביעה אל הרשומה הבאה ואל הרשומה הקודמת בטבעת.

טבעת מורכבת (Multiring) הינה אוסף של מספר טבעות פשוטות. כל רשומה יכולה להשתייך לטבעות שונות, ולדוגמא נבנה טבעת אחת הממיינת את כל רשומות העובד לפי שם וטבעת שניה הממיינת את כל הרשומות לפי גיל.

4.3 עצים (Trees)

מבנה עץ הינו **מבנה לא־לינארי** אשר מאפשר תצוגת **מבנה נתונים היררכי**. העץ מורכב מהירככיה של **צמתים** (Nodes) שבראשה הצומת הגבוה שנקרא **שורש** (Root). למעט השורש, במבנה עץ כל צומת קשור לצומת אחד ברמה יותר גבוהה, שיקרא **צומת־אב**. לכל צומת יש צומת אב אחד בלבד.

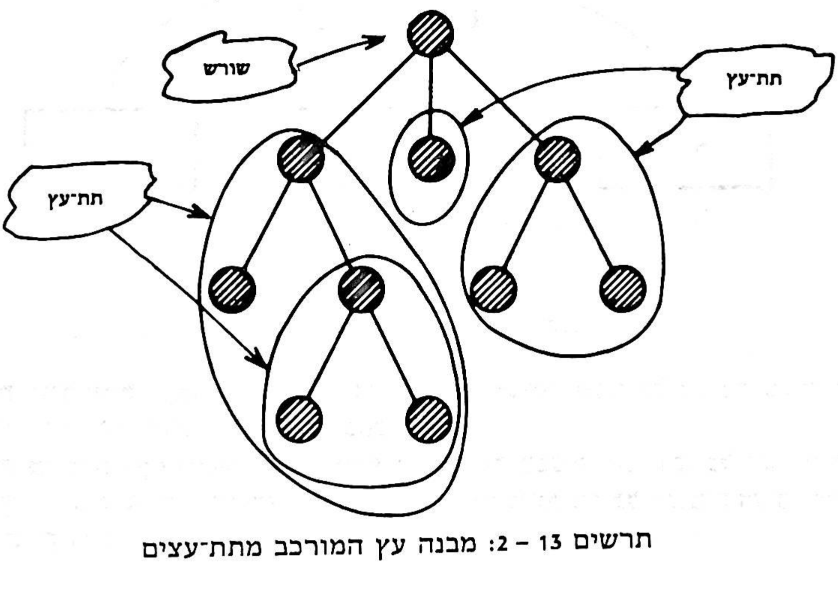


תרשים 12 – 2: מבנה עץ

הגדרה פורמלית של מבנה עץ מופיעה אצל Knuth (ראה ביבליוגרפיה).
עץ הינו קבוצה סופית T של אחד או יותר צמתים המקיימים:

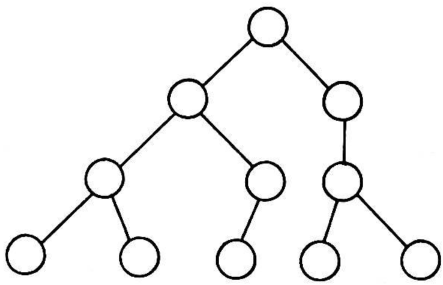
א. צומת אחד ייקרא **השורש** של העץ,

ב. שאר הצמתים יחולקו ל־$m\geqslant 0$ קבוצות זרות T1, T2...Tm כאשר כל אחת מהן יכולה להיות עץ נפרד. עצים אלו הם **תת־עצים** של מבנה העץ.



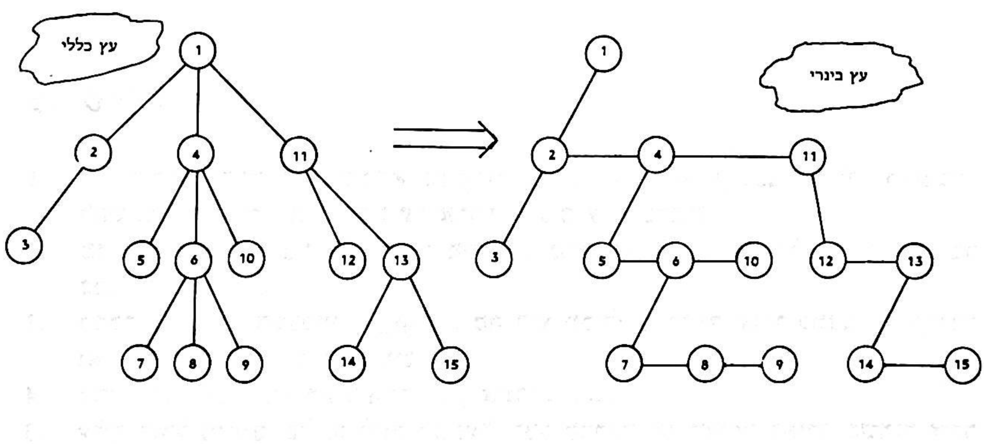
תרשים 13 – 2: מבנה עץ המורכב מתת־עצים

אינדקסים ואלגוריתמים לחיפוש מתאימים באופן טבעי למבנים של עצים ומכאן הענין הרב במבנים אלה. מבנה בעל ענין מיוחד הינו **מבנה עץ בינרי**. זהו עץ שבו לכל צומת שני ענפים לכל היותר. עץ בינרי פשוט קל ליישום ביחידות אחסון כי בכל צומת יש לשמור מקום לשני מצביעים בלבד. מצביע אחד מורה אל צומת ברמה נמוכה יותר בעץ הכללי ומצביע שני מורה אל צומת שכן באותה רמה הירכית.



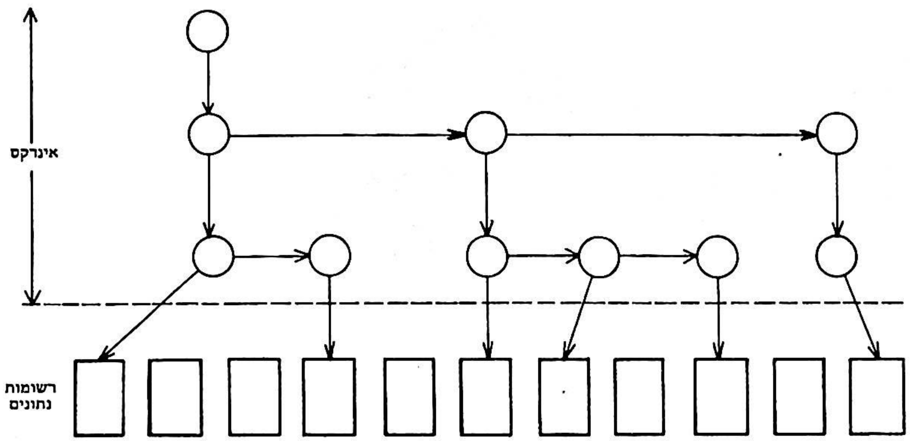
תרשים 14 – 2: עץ בינרי

כל עץ כללי ניתן להפוך לעץ בינרי. בזמן ההפיכה גדל העומק של העץ, או מספר הרמות שלו. העומק של העץ מבטא את מספר ההשוואות שיש לבצע כדי להגיע מהשורש לצומת ברמה הנמוכה ביותר. למרות הגידול בעומק העץ נוהגים לבצע את ההפיכה לעץ בינרי בגלל הנוחות שביישום השיטה.



תרשים 15 – 2: הפיכת עץ כללי לעץ בינרי

בתרשים 16—2 מוצג יישום של עץ בינרי עבור אינדקס ורשומות נתונים. החיפוש באינדקס פשוט ומהיר. על ידי רמות אינדקס אחדות ניתן לשפר את היעילות במידה רבה.



תרשים 16 – 2: אינדקס במבנה עץ בינרי

5. שאלות

1. הצג והסבר את ההבדלים בין "ארגון קבצים בסיסי" לבין "ארגון קבצים עילי." האם ניתן להשתמש במערכת המבוססת על ארגון קבצים עילי בלבד?
2. מהם השיקולים לקביעת מפתחות משניים? מדוע לא לקבוע את כל השדות ברשומה כמפתחות משניים?
3. הסבר את מושג המצביע (Pointer). מה הוא מכיל? מדוע יש צורך במצביעים קדימה ואחורה במבני נתונים מסויימים?
4. מהו ההבדל בין שרשרת הפוכה לבין אינדקס משני?
5. עליך לנהל נתונים של מחלקות המפעל. לכל מחלקה יש רשומה באורך משתנה אשר מכילה את שם המחלקה ואת מספרה, ונגררות עבור כל אחד מהעובדים במחלקה. הנתונים שדרושים עבור כל עובד הם שמו ומספר תעודת הזהות שלו וכל הפרויקטים שבהם עסק (נגררות). הנך מתבקש להציע מבנה נתונים לפתרון בעיית ניהול הנתונים באמצעות ארגון קבצים בסיסי וארגון קבצים עילי.

פרק ג

מודל הנתונים

1. מבוא
2. מושגי יסוד
3. מודל ישויות - קשרים
4. מודל הנתונים
5. המודלים הלוגיים העיקריים
6. סיכום
7. שאלות

1. מבוא

בפרק זה נציג את אחד המושגים המרכזיים במערכות DBMS — את **מודל הנתונים**. בניית מודל הנתונים הינה משימתו העיקרית של **מנהל בסיס הנתונים** בהקמת מערכת מידע ממוחשבת, מכיון שהמודל הינו הכלי **לניתוח הנתונים** שיש לכלול במערכת המידע. במערכות המידע שאינן מבוססות על DBMS הצטמצם תהליך הניתוח להבנת הקלטים והפלטים של המערכת ולבניית הקובץ אשר יכול לנהל נתונים אלה. הקובץ מיועד לרוב ליישום אחד מוגדר ולכן קל יחסית לבצע את תהליך הניתוח המוביל למבנה הנתונים הדרוש.

תהליך ניתוח הנתונים ותכנון בסיס הנתונים הינו מורכב יותר. בסיס הנתונים צריך לשרת מספר רב של יישומים ביעילות גבוהה ומבנה הנתונים צריך להיות גמיש כדי שניתן יהיה לבצע בו שינויים כתוצאה מדרישות מידע חדשות. מודל הנתונים הינו הכלי שבאמצעותו מתבצע ניתוח הנתונים.

התקשורת בין המשתמשים לבין מערכת המידע מתבצעת על סמך מודל כלשהו שהינו אוסף של כללים המשמשים לייצוג המציאות במערכות המידע. ניתוח המידע מתייחס לשני היבטים עיקריים:

— **ניתוח פונקציונלי**, שבו מנתח המערכת מבין ומתעד את התהליכים והפעילויות העיסקיות בארגון. אלה הם האירועים שמערכת המידע תצטרך לטפל בהם.

— **ניתוח נתונים**, שבו מנתח המערכת מבין ומתעד את מרכיבי מערכת המידע, המאפיינים שלהם והקשרים ביניהם. אלה הם הנתונים והקשרים שמערכת המידע תאחסן ותטפל.

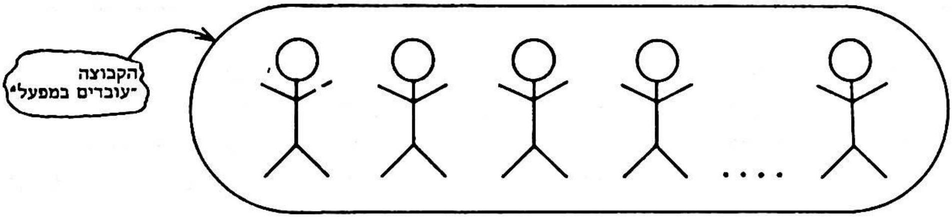
בהמשך נתרכז בתהליך ניתוח הנתונים ונציג את **מודל ישויות - קשרים** שבעזרתו נכיר טוב יותר את הדרך שבה אנשים מבינים את משמעות הנתונים. בהמשך נראה שניתן להתייחס לנתונים במספר רמות שונות — הרמה שבה אנשים מארגנים ומבינים את המציאות, הרמה שבה נותנים משמעות למציאות והרמה שבה משתמשים במודל הנתונים כדי לייצג את המציאות במערכות המידע. בסוף הפרק נציג את שלושת המודלים הנפוצים: המודל הרשתי, המודל ההירככי והמודל הטבלאי.

2. מושגי יסוד

בסעיף זה נציג מספר מושגי יסוד מתורת הקבוצות אשר קשורים לנושא מודל הנתונים. הצגתם תעשה בצורה לא פורמלית ותסתייע במספר רב של דוגמאות. הקורא אשר אינו מעוניין יוכל לעבור לסעיף 2.4 בהמשך.

2.1 קבוצה (Set)

הקבוצה מוגדרת **כאוסף של עצמים או אלמנטים**. אפשר לתאר את הקבוצה בשתי צורות: על ידי **הצגת כל האלמנטים** שלה או על ידי **הגדרת התכונות** המאפיינות את האלמנטים בקבוצה. נסמן את הקבוצה באותיות גדולות (A, B, ...) ואת האלמנטים באותיות קטנות (a, b, ...) בהתאמה.



A = עובדים במפעל = {משה, דוד, דן, ישראל, שאול, ...}

תרשים 1 – 3: ייצוג מלא של קבוצה

בתרשים 1—3 ניתן ייצוג מלא של קבוצה שבה האלמנטים הם העובדים במפעל. תיאור כל האלמנטים של הקבוצה אפשרי בקבוצות קטנות בלבד ולכן נעדיף את שיטת התיאור השנייה שמציגה את הקבוצה לפי התכונות של האלמנטים.

[a : a עובד במפעל] = עובדים במפעל = A

כלומר, הקבוצה A מכילה אלמנטים a המקיימים את התנאי ש־a הוא עובד במפעל.

קבוצה יכולה להיות סופית או אינסופית. **קבוצה סופית** מכילה **בדיוק** n אלמנטים שונים (n הוא מספר שלם חיובי). אם אינה ממלאת תנאים אלו היא תהיה **קבוצה אינסופית**. למשל, הקבוצה "עובדים במפעל" היא קבוצה סופית בעוד הקבוצה "מספרים חיוביים זוגיים" היא קבוצה אינסופית.

הקבוצה מאופיינת בכך שכל אלמנט מופיע בה פעם אחת בלבד וסדר האלמנטים בה חסר משמעות. לקבוצה יש גודל הנקבע על ידי מספר האלמנטים החברים בקבוצה.

העובדה שאלמנט מסוים **הינו חבר בקבוצה** תסומן בצורה הבאה:

$a \in A$, עובדים במפעל $\in$ משה

העובדה שאלמנט מסויים **אינו חבר בקבוצה** תסומן בצורה הבאה:

$\chi \notin A$, עובדים במפעל $\notin$ יצחק

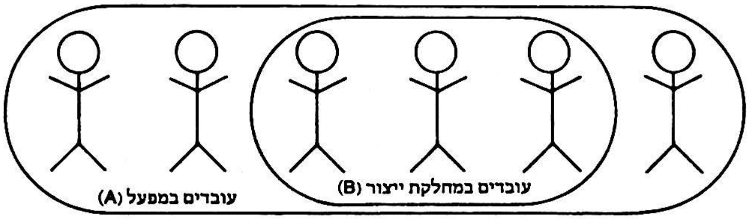
קבוצה ריקה (Null Set) אינה מכילה אלמנטים. תתכן למשל קבוצה ריקה "עובדים במחלקת צביעה" כאשר המחלקה נמצאת בשלבי הקמה וטרם נקבעו העובדים שלה. במשך הזמן יצטרפו אליה אלמנטים (עובדים) והקבוצה תפסיק להיות ריקה.

ניתן להגדיר **קבוצה חלקית** (Subset) שבה כל אלמנט החבר בקבוצה החלקית הינו חבר גם בקבוצה המלאה אבל לא להיפך. לדוגמא:

A = עובדים במפעל = [a :a עובד במפעל]

B = עובדים במחלקת ייצור = [b :b עובד במחלקת ייצור]

הקבוצה B הינה קבוצה חלקית ל־A מכיון שלא כל העובדים במפעל עובדים במחלקת ייצור. מקובל לסמן עובדה זו בצורה הבאה: $B \subset A$.



תרשים 2 – 3: קבוצה חלקית B של A

קבוצות זרות הן קבוצות נפרדות שאינן חופפות. הגדרה של קבוצות נפרדות אינה מונעת מצב בעתיד שבו הן תתלכדנה או תחפופנה זו את זו באופן חלקי או מלא.

לדוגמא, נגדיר שתי קבוצות של תלמידים בכתה אחת: קבוצת לימוד מתמטיקה וקבוצת לימוד פיסיקה. כלומר, לפנינו קבוצה אחת (הכתה) שבה שתי קבוצות חלקיות. עתה נניח שהקבוצה החלקית של לימוד פיסיקה מתבטלת, וכל התלמידים ילמדו מתמטיקה בלבד. במצב החדש יש התלכדות של הקבוצה החלקית של לומדי מתמטיקה ושל הקבוצה העיקרית. אפשר להניח אפשרות אחרת שבה תלמידים יחליטו להשתתף הן בלימודי המתמטיקה והן בלימודי הפיסיקה. תלמידים אלו ישתתפו בשתי קבוצות חלקיות, שמלכתחילה היו **זרות** זו לזו, כלומר ללא כל חפיפה.

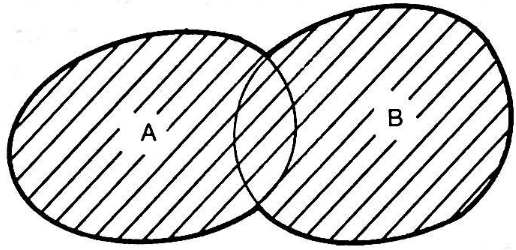
כדוגמא נוספת נגדיר שתי קבוצות של פריטים: פריטים המיוצרים במפעל ומשמשים למכירה בלבד ופריטים הנקנים כחומרי גלם מיצרנים אחרים ומשמשים לייצור בלבד. לכאורה יש כאן מצב ברור של שתי קבוצות חלקיות זרות אך עם זאת יתכן שבעתיד לקוח כלשהו של המפעל ירצה לקנות מהמפעל חומר גלם כלשהו. במקרה זה פריטים מהקבוצה החלקית של חומרי־גלם ימצאו גם בקבוצה החלקית של פריטים למכירה.

2.2 פעולות בקבוצות

יחסי הגומלין בין קבוצות אפשר לבטא במספר פעולות עיקריות כפי שנפרט בהמשך.

איחוד (Union): איחוד שתי קבוצות A ו־B יסומן בצורה AUB. האיחוד יוצר קבוצה חדשה שמכילה את כל האלמנטים השייכים לקבוצה A **או** לקבוצה B.

$$A \cup B = \{\chi : \chi \in A \text{ או } \chi \in B\}$$



תרשים 3 – 3: AUB הוא השטח המקווקו

לדוגמא, נגדיר שתי קבוצות של עובדים:

A = עובדים במחלקת ייצור = [ישראל, דוד]

B = עובדים במחלקת תכנון = [שאול, ישראל, דן]

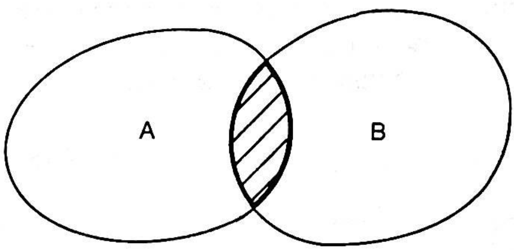
האיחוד בין שתי הקבוצות יהיה:

AUB = עובדים במחלקות ייצור ותכנון = [ישראל, שאול, דן, דוד]

נשים לב לכך שעובד בשם "ישראל" נמצא בשתי המחלקות. בקבוצת האיחוד הוא מופיע פעם אחת **בלבד**.

חיתוך (Intersection): חיתוך שתי קבוצות A ו־B, (A∩B) יוצר קבוצה חדשה שמכילה את כל האלמנטים השייכים לקבוצה A **וגם** לקבוצה B.

$$A \cap B = \{\chi : \chi \in A \text{ ו } \chi \in B\}$$



תרשים 4—3: A∩B הוא השטח המקווקו

אם נשתמש בדוגמא הקודמת, קבוצת החיתוך תכיל אלמנט אחד, את העובד "ישראל", אשר מופיע בקבוצה A וגם בקבוצה B.

$A \cap B$ = עובדים בשתי המחלקות = [ישראל]

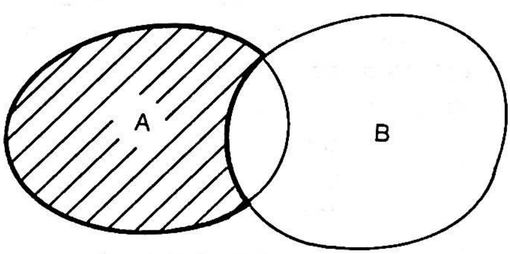
אם $A \cap B = O$ החיתוך יוצר קבוצה ריקה, כלומר שתי הקבוצות הן **זרות** (Disjoint).

הפרש (Difference): ההפרש בין קבוצה A ו־B ‏(A—B) יוצר קבוצה חדשה שמכילה את האלמנטים השייכים לקבוצה A **ואינם** שייכים לקבוצה B.

$$A - B = [\chi : \chi \in A,\ x \notin B]$$

לדוגמא נגדיר קבוצת הפרש בין העובדים במחלקת ייצור לבין העובדים במחלקת תכנון. כלומר קבוצה אשר תכיל את העובדים במחלקת הייצור בלבד ואשר אינם משתתפים במחלקת התכנון. ולפי הדוגמא הקודמת יהיה זה העובד "דוד".

$A - B$ = עובדים במחלקת ייצור בלבד = [דוד]



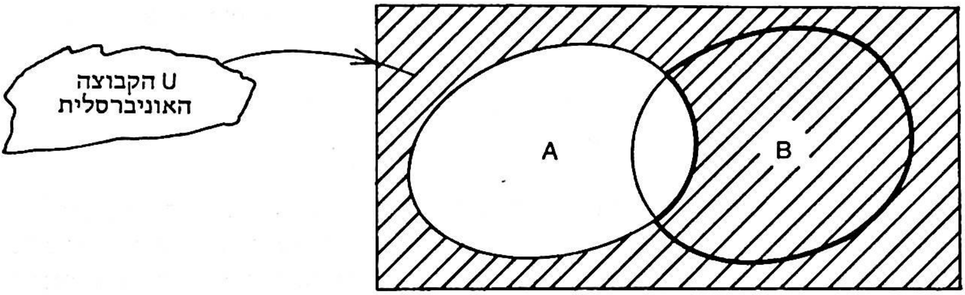
תרשים 5 – 3: A—B הוא השטח המקווקו

משלים (Complement): המשלים של קבוצה A, שיסומן A^c יוצר קבוצה חדשה שמכילה אלמנטים שאינם שייכים לקבוצה A. יש להגדיר קבוצה אוניברסלית U אשר מכילה את כל האלמנטים האפשריים בהקשר הנדון.

$$A^C = [\chi : \chi \in U,\ \chi \notin A]$$

קבוצה U בדוגמא היא קבוצת כל העובדים במפעל. קבוצת המשלים של העובדים של מחלקת הייצור היא כל העובדים שאינם שייכים למחלקה זו.

A^c=עובדים שאינם שייכים למחלקת ייצור=[שאול, דוד]



תרשים 6 – 3: A^c הוא השטח המקווקו

קבוצת מכפלה (Product Set): להגדרת קבוצת מכפלה דרושה הגדרה של **זוג מסודר** (Ordered Pair) אשר מורכב משני אלמנטים, a ו־b. כאשר a הוא האלמנט הראשון ו־b הוא האלמנט השני נסמן את הזוג בצורה (a,b). אפשר להגדיר סדר של מספר כלשהו של אלמנטים. למשל רביעיה מסודרת תרשם כך: (a, b, c, d).

נניח ש־A ו־B הן שתי קבוצות. קבוצת המכפלה או **המכפלה הקרטזית** (Cartesian Product) של A ו־B, שתסומן A×B, מכילה את קבוצת כל הזוגות המסודרים (a, b) כך שהאלמנט הראשון בא מקבוצה A והאלמנט השני בא מקבוצה B.

$$A \times B = [(a,b) : a \in A, b \in B]$$

לדוגמא נגדיר קבוצה של מנהלים וקבוצה של עובדים.

A = מנהלים = [דן, דוד]

B = עובדים = [שאול, ישראל, משה]

המכפלה הקרטזית בין שתי קבוצות אלה תגדיר את כל האפשרויות של יחסים מנהל—עובד.

A×B = [(שאול, דן), (שאול, דוד), (ישראל, דן), (ישראל, דוד), (משה, דן), (משה, דוד)]

ניתן להגדיר מכפלה קרטזית של קבוצה עם עצמה. אם קבוצה A מורכבת מאלמנטים [y, x], הקבוצה A×A תהיה:

$$A \times A = [(y,y), (y,x), (x,y), (x,x)]$$

ניתן להגדיר את פעולת המכפלה על מספר כלשהו של קבוצות. לדוגמא, המכפלה הקרטזית של קבוצות A, B, C שתסומן A×B×C מורכבת מכל השלשות המסודרות, כך ש:

$$A \times B \times C = [(a, b, c): a \in A, b \in B, c \in C]$$

לדוגמא נתונות קבוצות:

A= [a, b, c] B=[x, y] C=[z, k]

המכפלה הקרטזית שלהן תהייה:

A×B×C = [(a,x,z), (b,x,z), (c,x,z),
(a,y,z), (b,y,z), (c,y,z),
(a,x,k), (b,x,k), (c,x,k),
(a,y,k), (b,y,k), (c,y,k)]

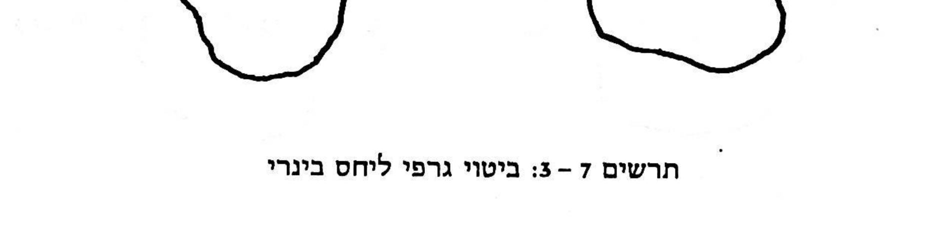
2.3 יחסים (Relations)

היחס הינו אחד המושגים החשובים במודלים של נתונים. הוא מתאר קשר בין אלמנטים של אותה קבוצה או בין אלמנטים של קבוצות שונות. לדוגמא בנקודת זמן כלשהי, רק חלק מבין כל הזוגות האפשריים (עובד, מנהל) מייצגים קשר קיים בין מנהל לעובד.

היחס הבינרי (Binary Relation) הינו **קבוצה חלקית** לקבוצת המכפלה הקרטזית של שתי קבוצות A ו־B למשל. הביטוי המתמטי שלו יהיה:

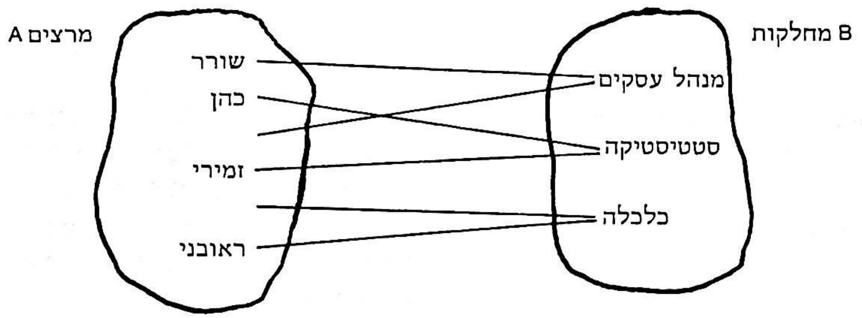
$$R(A,B) \subseteq A \times B$$

סדר הופעת האלמנטים ביחס בינרי הוא חשוב. בדוגמא שבתרשים 3—7 האלמנט הראשון יהיה תמיד מתוך קבוצת המנהלים והאלמנט השני יהיה תמיד מתוך קבוצת העובדים. בדוגמא זו לא קיימים הקשרים "דן־משה" ו"דוד־שאול".



תרשים 7 – 3: ביטוי גרפי ליחס בינרי

נציג דוגמא נוספת ליחס בינרי ונגדיר לשם כך שתי קבוצות: קבוצה A היא קבוצת כל המרצים באוניברסיטה וקבוצה B היא קבוצת המחלקות השונות באוניברסיטה. המכפלה הקרטזית $A \times B$ מכילה את כל הזוגות המסודרים האפשריים בין מרצה לבין המחלקה כאשר האלמנט הראשון יהיה מרצה והאלמנט השני יהיה מחלקה. במציאות קיים קשר בין המרצים והמחלקות אך הקשר אינו מקיף את כל המקרים מכיון שרק חלק מהמרצים מלמדים במחלקה מסויימת. היחס הבינרי הקושר בין שני האלמנטים הינו קבוצה חלקית למכפלה הקרטזית.



תרשים 8 – 3: יחס בין מרצה למחלקה

יחס הינו הרחבה של מושג היחס הבינרי. **יחס במימד** n הינו קבוצה חלקית למכפלה הקרטזית של n קבוצות. יחס במימד 4 המתאר מספר מאפיינים של נתוני עובד יוצג בצורה הבאה:

עובדים במפעל = (שם עובד, גיל, מקצוע, שנות נסיון מקצועי)

כל אחד מהמאפיינים הינו למעשה קבוצה, כמו קבוצת שמות של העובדים, קבוצת כל הגילים, קבוצת המקצועות ועוד. המכפלה הקרטזית של ארבעת הקבוצות תהיה קבוצה גדולה מאד כי היא מייצגת את כל האפשרות בהתאם למספר האלמנטים בכל אחת מהקבוצות. יחס במימד 4 הינו קבוצה חלקית למכפלה הקרטזית והוא יכיל אלמנטים כמספר העובדים במפעל. נוח מאד לייצג את היחס בצורת טבלה.

מאפיינים

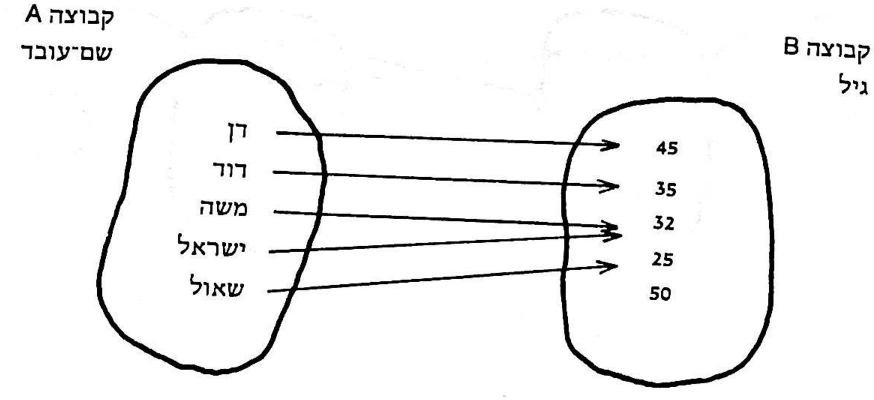
שם עובד	גיל	מקצוע	נסיון מקצועי
דן	45	מהנדס ייצור	8
דוד	35	טכנאי ייצור	4
משה	32	חרט	5
ישראל	32	טכנאי מכונות	3
שאול	25	מסגר	3
ראובן	39	שרטט	4

יחס

תרשים 9 – 3: ייצוג יחס במימד 4 על־ידי טבלה

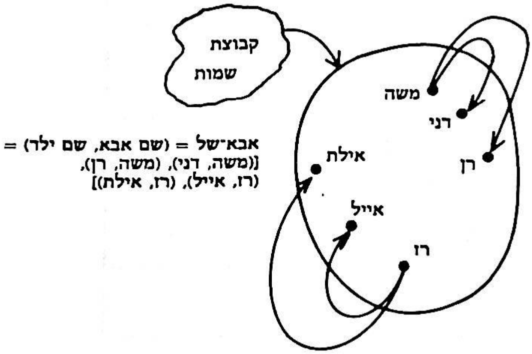
2.4 פונקציות (Functions)

פונקציה הינה מקרה מיוחד של קשר בין שתי קבוצות. נניח שלכל אלמנט בקבוצה A קושרים בדיוק אלמנט אחד מתוך קבוצה B. שני אלמנטים מקבוצה A יכולים להיות קשורים לאלמנט אחד מקבוצה B, ויתכנו אלמנטים ללא כל קשרים שהם. האוסף f של כל הקשרים האלה יקרא "פונקציה מ־A לתוך B" ויסומן: $f : A \rightarrow B$ ניתן להגדיר את הפונקציה גם כקבוצה חלקית של המכפלה הקרטזית $A \times B$ שבה כל אלמנט מתוך A מופיע פעם אחת בלבד.



תרשים 10 – 3: פונקציה בין שם עובד וגילו

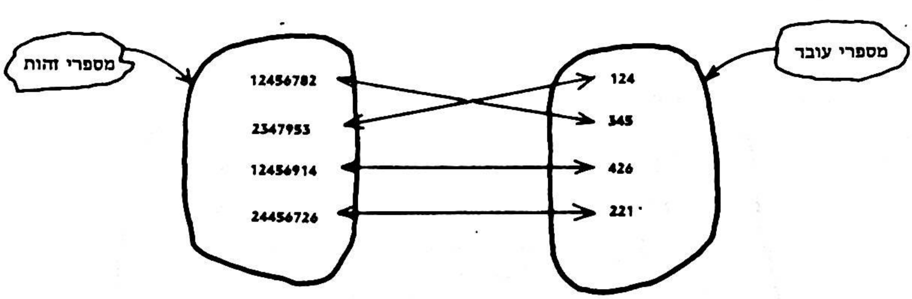
הפונקציה מגדירה את הקשר או את **המיפוי** (Mapping) בין אלמנטים בשתי קבוצות שונות או באותה קבוצה. הקשר בין האלמנטיים יכול להיות חד—חד־ערכי (1:1), חד—רב־ערכי (1:N) או רב—רב־ערכי (N:M).



אבא	שמות ילדים
משה	דני, רן
רז	אייל, אילת

תרשים 11 – 3: צורות שונות לייצוג קשרים

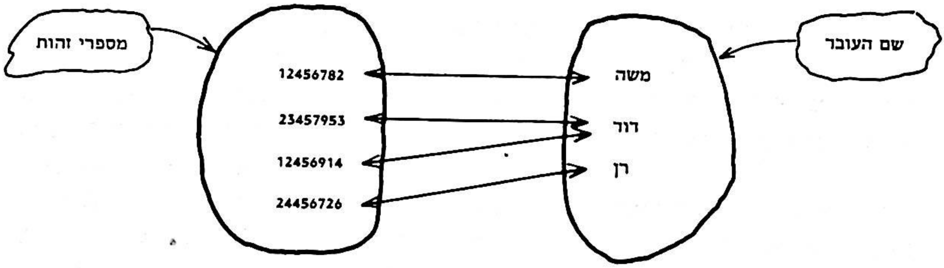
קשר חד – חד־ערכי (1:1) הוא הקשר הפשוט ביותר הקיים בין האלמנטים. לכל אלמנט בקבוצה אחת יש התאמה לאלמנט אחד בקבוצה השניה ולהיפך, כמו לדוגמה הקשר בין קבוצת מספר הזיהוי של העובד לבין קבוצת מספרי העובד. המשמעות היא שלכל מספר זהות מקבוצה אחת מתאים מספר עובד אחד בלבד מהקבוצה השניה ולהיפך. מקובל לסמן קשר כזה בקו שחיצים בשני קצותיו ↔.



תרשים 12 – 3: קשר חד – חד־ערכי

בצורה פורמלית נאמר שקיים קשר 1:1 אם קיימות שתי קבוצות A ו־B שניתן להגדיר עבורן שתי פונקציות f:A→B, f:B→A. כזכור, הפונקציה מאפשרת לאלמנט הראשון להופיע פעם אחת בלבד.

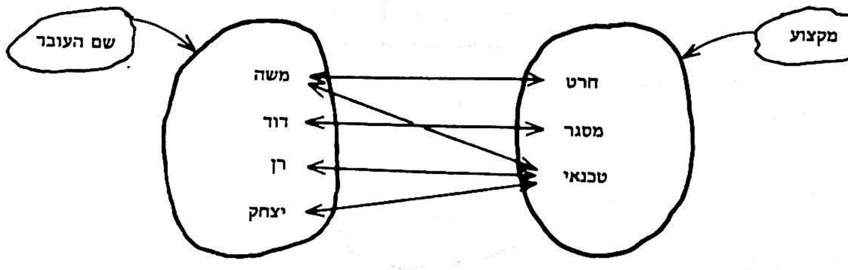
קשר חד – רב־ערכי (1:N) מאפשר שלכל אלמנט בקבוצה אחת תהייה התאמה למספר אלמנטים בקבוצה השניה אבל לא להיפך. כלומר, בין אלמנטים בקבוצה השניה לאלמנטים בקבוצה הראשונה קיימת התאמה חד־ערכית. הקשר בין קבוצת שמות העובדים לבין קבוצת מספרי הזיהוי שלהם הוא מסוג זה. קשר רב־ערכי קיים בין מספרי הזיהוי לבין השמות מכיון שניתן להתאים לכל שם מספרי זיהוי אחדים. הקשר בין השמות לבין מספרי הזיהוי הוא חד־ערכי מכיון שלכל שם יש מספר זיהוי אחד בלבד. מקובל לסמן קשר זה בקו שבצידו האחד חץ אחד ובצידו האחר ישנם שני חיצים ←←→.



תרשים 13 – 3: קשר חד – רב־ערכי

בצורה פורמלית נאמר שקיים קשר 1:N אם קיימות שתי קבוצות A ו־B וניתן להגדיר עבורן רק פונקציה אחת, f:A→B או f:B→A.

קשר רב – רב־ערכי (N:M) מציין שלכל אלמנט בקבוצה אחת יש התאמה למספר אלמנטים בקבוצה השניה ולהיפך. הקשר בין קבוצת שמות העובדים לבין קבוצת המקצועות שלהם הוא קשר רב—רב־ערכי. יתכנו מספר עובדים בעלי שם זהה עם מקצועות שונים וכן יתכנו מספר עובדים בעלי שם שונה שיש להם מקצוע זהה. מקובל לסמן קשר זה בקו ששני חיצים בכל אחד מקצותיו: ←←→→.



תרשים 14 – 3: קשר רב – רב־ערכי

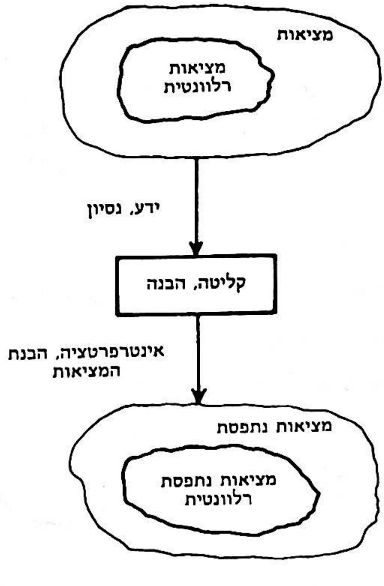
בדוגמא המופיעה בתרשים 14—3 ניתן לראות ששני עובדים ששמם "משה" הם בעלי מקצועות "חרט", "טכנאי", ושלשה עובדים בעלי המקצוע "טכנאי" נושאים שמות שונים "משה", "רן", ו"יצחק".

בצורה פורמלית נאמר שקיים קשר N:M אם קיימות שתי קבוצות A ו־B ולא ניתן להגדיר אף אחת מהפונקציות f:A→B או f:B→A.

3. מודל ישויות – קשרים

מטרת תהליך ניתוח הנתונים היא לבנות מבנה נתונים שמתאים לכל היישומים בארגון, והוא צריך להיות גמיש למדי כדי לאפשר ביצוע שינויים ללא זעזוע מערכת המידע. תוך כדי הניתוח מזהים ומסווגים את הנתונים הקיימים בארגון ואשר דרושים למערכת המידע. מטרת תהליך זה היא לבנות שיטת התייחסות זהה ואחידה לכל הנתונים, וזהו **מודל הנתונים של הארגון** (Organization Data Model).

התוצר של הניתוח והלימוד הוא ארגון נתונים וקשרים שבמציאות לצורה שמובנת ונתפסת על ידי האנשים שמתכננים את מערכת המידע ונעזרים בשירותיה. בדרך כלל מערכת המידע עוסקת רק בחלק מהמציאות, בזו אשר רלוונטית לארגון. בעקרון היינו רוצים לנהל במערכת המידע נתונים על כל המציאות הרלוונטית, אך בגלל אילוצים של כדאיות כלכלית וכמויות נתונים שיש לאחסן מתבצע תהליך לצמצום נוסף של המציאות הרלוונטית. בסופו של התהליך מכילה מערכת המידע רק את הנתונים הדרושים לה לפי דרישתם והבנתם של המתכננים.



תרשים 15 - 3: הפיכת המציאות למציאות נתפסת

כדי להסביר את משמעות התכנון נשתמש בדוגמא של ניתוח מערכת המידע במפעל כלשהו. המציאות הרלוונטית מבחינת המפעל מורכבת מהקונים הפוטנציאליים של המוצר, ספקים של חומרי גלם, מכונות, פועלים העובדים במפעל, תהליכי עיבוד החומר והפיכתו למוצר, ועוד. למרות הרשימה הארוכה זהו רק חלק קטן מהמציאות כולה. עקרונית, הארגון היה מעוניין לקבל מידע רב על נושאים מגוונים כמו מצב פוליטי שיכול להשפיע על המכירות, מדיניות מחירים של הממשלה ושל המתחרים, המוצרים המתחרים ומידע על היצרנים שלהם. כל אלה הם חלק מהמציאות הרלוונטית אבל יתכן שלא יוכנסו למערכת המידע מסיבות שונות כמו עלות גבוהה לאיסוף הנתונים, כמויות נתונים גדולות שיש לאחסן במערכת המידע, קושי בהגדרה ובאחסון של נתונים איכותיים ועוד. בגלל קשיים אלו חל תהליך של צמצום נוסף של המציאות הנתפסת.

מודל **ישויות־קשרים** (Entity-Relationship Model) הינו כלי אינטלקטואלי אשר מאפשר בניית מודל תפישתי של המציאות אשר משוחרר מכל היבטים הקשורים למחשבים. מטרתו היא לאפשר הבנת המציאות וקביעת מושגים מקובלים ומוסכמים לבניית יחידות לוגיות המייצגות את המציאות ומאפשרות למנתח המערכות לטפל בה בצורה סימבולית.

3.1 מרכיבי העולם הממשי

העולם הממשי מורכב מאלמנטים אשר בונים את המציאות. מודל ישויות־קשרים משתמש בשני מרכיבים להבנת המציאות: **ישות** ו**קשר** אשר מבטא קשר בין ישויות. לכל אחד משני מרכיבים אלה יש מספר מאפיינים שיוצגו בהמשך.

3.1.1 ישות (Entity)

המציאות מורכבת מאוסף של ישויות. עקרונית, ניתן להתייחס לכל דבר כאל ישות, אך זהו תחום אינסופי ולכן קביעת הישות במערכת המידע נובעת מהצורך של הארגון לנהל אודותיה נתונים מסויימים. ישות יכולה להיות מוחשית או לא־מוחשית, כפי שנראה בדוגמאות שלהלן:

— **עצם** כמו מכונית, בית, מכונה במפעל, פריט במלאי;
— **גוף־חי** כמו עובד במפעל, סטודנט באוניברסיטה, חולה בבית חולים;
— **מושג מופשט** כמו תקופת זמן, תאריך לידה, שם של איש, ארגון עסקי, קורס;
— **אירוע** כלשהו כמו שעת כניסת העובד למפעל, אספקת מוצר בעוד כחודשיים.

לישות יש זכות קיום משל עצמה וניתן להבדיל בינה לבין ישויות אחרות. המנתח מגביל את עצמו לאותן ישויות אשר רלוונטיות לצורך הקמת מערכת המידע מכיון שאין אפשרות לאחסן את כל המידע הקיים. הישויות הרלוונטיות נקבעות בהתאם לאופי הארגון ומטרותיו. מערכת המידע תאחסן נתונים **אודות הישות** ולא את **הישות עצמה**.

מאפיין (Characteristic, Property)

לכל ישות יש מאפיין אחד או יותר. לדוגמא:
— לישות **מכונית** יש מאפיינים כגון "מספר רישוי", "שנת ייצור", "דגם" ועוד,

- לישות **עובד במפעל** יש מאפיינים כגון "שם העובד", "מספר הילדים שלו", "תאריך תחילת העבודה במפעל", "מחלקה בה הוא עובד" ועוד,
- לישות **אספקת מוצר** יש מאפיינים כגון "תאריך האספקה", "כמות במשלוח", "שם הנמען" ועוד.

המאפינים מאפשרים את זיהוי, אפיון, סיווג וקישור הישויות השונות. אחת הבעיות הקשות ביותר של מנתח המערכות היא הקביעה של המאפיינים של היישות.

במערכת מידע על עובדי המפעל למשל נשתמש לייצוג היישות "**עובד**" במאפיינים שונים ביניהם "שם העובד", "תאריך תחילת העבודה", "נסיון מקצועי", "כתובת" ועוד. יתכן שלא נבחר במאפיין "סיווג בטחוני של העובד" מכיון שאינו רלוונטי למערכת המידע המתוכננת. אך יתכן שלאחר זמן מה המפעל יזכה במכרז לייצור ציוד בטחוני ויהיה זקוק למידע אודות הסיווג הבטחוני של עובדיו. במקרים מסויימים יש על כן לאגור מידע גם אם לכאורה הוא לא רלוונטי למערכת המידע המתוכננת אך הוא עשוי להיות מועיל או דרוש בעתיד.

בעייה אחרת היא ההחלטה מהי ישות ומהו מאפיין כי האבחנה ביניהם הינה סוביקטיבית לחלוטין ותלוייה בנסיבות. במערכת מידע אוניברסיטאית יש חשיבות רבה מאד לישות "**קורס**" שכמה ממאפייניה הם "שם הקורס", "שם המרצה", "מספר נקודות זכות שהוא מקנה", "משך הקורס" ועוד. לעומת זאת, במערכת מידע על עובדי המפעל יהיה לישות **עובד** מאפיין של "שם הקורס בו השתתף". דוגמא נוספת יכולה להיות מערכת מידע בנושא פיקוח תחבורתי של טיב הכבישים. מערכת זו תנהל ישות עצמאית **רחוב** עם מספר מאפיינים, ולעומת זאת במערכת המידע על עובדי המפעל לישות "עובד" יש מאפיין "רחוב בו גר".

האבחנה בין ישות למאפיין יכולה להתבצע רק לאחר הבנת המציאות הרלוונטית למערכת המידע המתוכננת והגדרתה.

ערך המאפיין (Characteristic/Property Value)

כל מאפיין יכול לקבל ערך כלשהו המייצג אותו. לדוגמא,
- המאפיין **יצרן המכונית** יכול לקבל אחד מתוך הערכים פיאט, פורד, סוברו,
- המאפיין **מין** יכול לקבל אחד מתוך שני הערכים זכר, נקבה,
- המאפיין **מספר זהות של העובד** יכול לקבל אחד מתוך הערכים האפשריים של מספרי הזהות של העובדים במפעל.

תחום ערכים (Domain)

תחום הערכים הינו אוסף כל הערכים האפשריים שמאפיין מסויים יכול לקבל, לדוגמא:
- תחום הערכים של המאפיין **יצרן המכונית** הוא אוסף כל היצרנים של מכוניות בעולם.
- תחום הערכים של המאפיין **מספר זהות** הוא אוסף כל המספרים החיוביים בני 8 ספרות שנקבעו על-ידי משרד הפנים כמספר זהות של התושבים,
- תחום הערכים של המאפיין **כתה בבית-ספר עממי** הוא המספרים השלימים מ-1 ועד 8.

מאפיינים שונים יכולים לקבל את הערכים שלהם מתוך אותו תחום ערכים. למשל המאפיין "צבע שיער" וגם המאפיין "צבע עינים" מקבלים את הערכים שלהם מתוך תחום הצבעים האפשריים. למרות שהערכים יכולים להיות זהים המשמעות שלהם שונה לחלוטין.
תחום הערכים שהמאפיין יכול לקבל קובע את צורת הייצוג שלו במערכת המידע. לדוגמא,

המאפיין "מספר זהות" ייוצג במערכת על־ידי מספר בן 8 ספרות בעוד המאפיין "שם עובד" ייוצג על ידי נתון אלפא־ביתי בן 30 תוים.

מפתח היישות (Entity Key)

נגדיר את מפתח הישות כמאפיין המזהה את הישות באופן חד־משמעי. לא תמיד יש לישות מאפיין כזה ולכן מקובל להוסיף מאפיין מלאכותי המשמש למטרות זיהוי בלבד. נשתמש במספר דוגמאות:

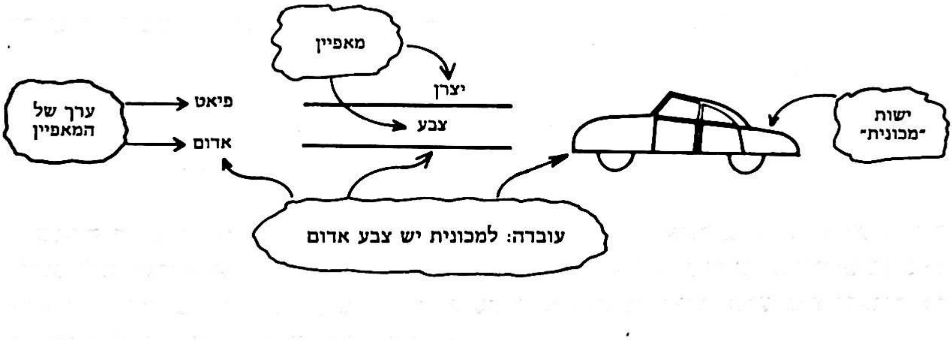
- הישות **מכונית** תזוהה על־ידי המאפיין המלאכותי "מספר רישוי" מכיון שאין כל מאפיין אחר שיכול לזהות את הישות בצורה חד־משמעית.
- הישות **סטודנט** תזוהה על־ידי המאפיין המלאכותי "מספר סטודנט" כי אין כל מאפיין אחר אשר יכול לזהות את הישות בצורה חד־משמעית. יתכנו למשל סטודנטים שונים עם אותו שם. אפשר למשל להשתמש במאפיין "מספר זהות" שהוא חד־משמעי אך אינו יכול לסווג את הישות לפי דרישות וצרכי רישום הסטודנטים.

המפתח הוא **מפתח פשוט** אם הוא מורכב ממאפיין אחד, והוא יהיה **מפתח מורכב** אם הוא מכיל מספר מאפיינים אשר משמשים יחדיו לזיהוי הישות.

עובדה (Fact)

העובדה יוצרת קשר בין המאפיין לבין הערך שלו כי היא מגדירה את הישות, את שם המאפיין ואת הערך שלו. נציג דוגמאות אחדות:

- "יצרן המכונית הוא מפעלי פיאט": זוהי עובדה אשר יוצרת קשר בין המאפיין "יצרן המכונית" והערך "פיאט" שהמאפיין יכול לקבל.
- "שם העובד הוא משה": זוהי עובדה אשר יוצרת קשר בין המאפיין "שם העובד" והערך "משה" שהמאפיין יכול לקבל.



תרשים 16 – 3: קשר בין ישויות, מאפיינים וערכי מאפיינים

3.1.2 קשר (Relationship)

הקשר מגדיר יחס כלשהו בין מרכיבי המציאות, ישויות או תכונות. לדוגמא,

— "משה לומד באוניברסיטה העברית בירושלים": קשר בין הישות "עובד" כאשר הערך של המאפיין "שם העובד" הוא משה, לבין הישות "מוסדות להשכלה גבוהה" כאשר הערך של המאפיין "שם המוסד" הוא האוניברסיטה העברית בירושלים.
— "פריט 129 מורכב מפריטים 592, 924": קשר בין שתי ישויות מאותו סוג "פריט".
— "עובד מספר 145 שמו משה": קשר בין המאפיין "שם עובד" לבין המאפיין "מספר העובד". שני המאפיינים שייכים לישות אחת "עובד".

הקשר קובע את השמות של המרכיבים הקשורים ואת שם הקשר. בדוגמא הראשונה למשל שם הקשר הוא "לומד" והוא קושר את הישויות "עובד" ו־"מוסד להשכלה גבוהה". הקשרים נבדלים זה מזה כפי ששונים ומגוונים המרכיבים הקשורים באמצעותם, אך בכל זאת ניתן לסווג אותם במספר קבוצות עיקריות.

סוג ההתאמה: ההתאמה יכולה להיות מסוג 1:1, 1:N, N:M.

האלמנטים המשתתפים בקשר: האלמנטים המשתתפים בקשר יכולים להיות מאפיינים שונים של אותה ישות או של ישויות שונות וכל סוגי ההתאמה יכולים להתקיים. קשר בין ישויות הוא למשל הקשר בין העובד למחלקה שבה הוא עובד. קשר זה הוא חד—רב־ערכי כי העובד יכול לעבוד רק במחלקה אחת בעוד שבמחלקה אחת יכולים לעבוד מספר עובדים. ניתן להגדיר גם קשר בין אלמנטים שונים של אותה ישות כמו הקשר בין מנהל לעובדים שעליהם הוא ממונה.

הכרחיות הקשר: קיימים קשרים הכרחיים כמו למשל הקשר בין העובד למחלקה. קשר בין עובד וילדים הוא **אופציונלי** כי לא לכל עובד יש ילדים. קשרים אחרים אינם נחוצים כלל כי אין להם כל משמעות במערכת המידע שבה דנים.

תלות בזמן: קשר יכול להשתנות עם הזמן. קשר עם התאמה 1:1 בטווח הקצר יכול להפוך בטווח הארוך לקשר 1:N, כמו למשל הקשר בין אדם וחשבון חסכון בבנק. בזמן נתון יש לו חשבון חסכון אחד אך במשך הזמן הוא פותח חשבונות חסכון נוספים.

קשרים המוציאים זה את זה: קשר אחד יכול למנוע את האפשרות לקיום קשר אחר. אם מחסן סחורות מיועד לחומרי גלם בלבד או לתוצרת מוגמרת בלבד לא יתכן פריט מסוים המאוחסן בשני המחסנים. הקשר בין הפריט למחסן התוצרת המוגמרת מוציא את האפשרות לקיום קשר בינו לבין מחסן חומרי הגלם.

3.2 ייצוג המציאות במודל

בסעיף קודם הצגנו את המרכיבים הבסיסיים של המציאות אשר מערכת המידע צריכה לייצג. לשם הגדרת המציאות במונחים של מערכת מידע נדרשת הקבצה של מרכיבים בסיסיים או הרחבה שלהם, ולכך מכוון סעיף זה. שיטה זו מקילה על מנתח המערכות להגדיר את המציאות ולייעל את התקשורת עם אחרים.

קבוצת ישות (Entity set)

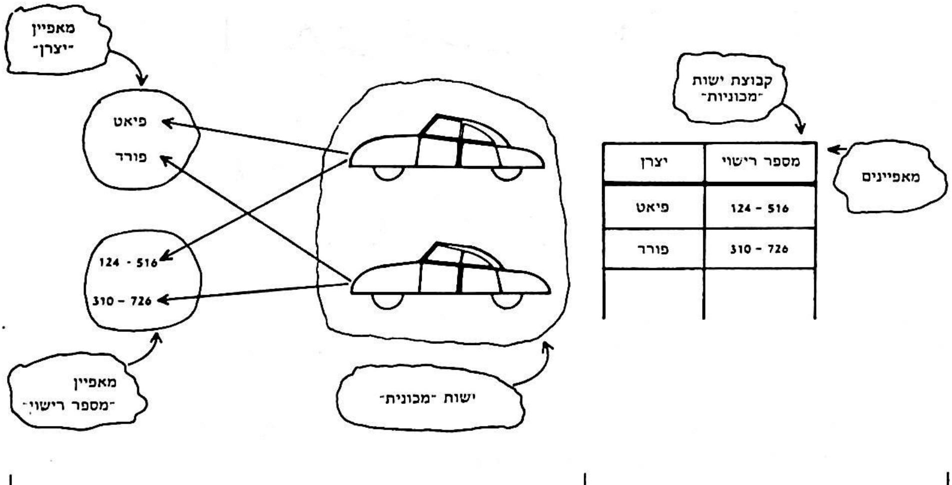
קבוצת הישות הינה אוסף **המופעים** (Occurrences) של כל הישויות שיש להם את אותם המאפיינים. בפועל, קבוצת הישות הינה מבנה לוגי המאפשר למנתח המערכות להתייחס אל המציאות מבלי שיהיה עליו להכיר את כל הישויות הממשיות ולטפל בהן. לאחר שנקבעו למשל מאפיינים רלוונטיים שיש לנהל עבור ישות **"עובד"** ניתן לטפל במבנה הלוגי "עובדים" מבלי להכיר את העובדים במפעל.

— לישות **"עובד במפעל"** נבנה קבוצת ישות "עובדים" המייצגת את כל העובדים במפעל, בקבוצה יהיה לכל ישות, כלומר עובד, מופע אחד;
— לישות **"מכונית"** נבנה קבוצת ישות "מכוניות במפעל" המייצגת את כל המכוניות הנמצאות בבעלות המפעל;
— לישות **"קבלת חומר מספק"** נבנה קבוצת ישות "אספקות מתוכננות" המייצגת את כל המשלוחים הצפויים.

ישות אחת יכולה להשתתף במספר כלשהו של קבוצות שונות. הישות **"עובד"** יכולה להשתייך לקבוצת "עובדים במפעל", לקבוצה "בוגרי מוסדות להשכלה גבוהה", ולקבוצה "מנהלי עבודה".

תכונה (Attribute)

קבוצת הישות היא בעלת משמעות כאשר היא מתוארת באמצעות תכונותיה. התכונה הינה האינטרפרטציה של מאפיין הישות הקיים במציאות. כמו קבוצת הישות "עובדים במפעל" אשר מתוארת באמצעות תכונות כגון "שם העובד", "כתובת העובד". זוהי הצגה של העולם האמיתי, המציאות, שבו לכל עובד יש מאפיינים שביניהם "שם העובד" ו־"כתובת".



יצרן	מספר רישוי
פיאט	124 - 516
פורד	310 - 726

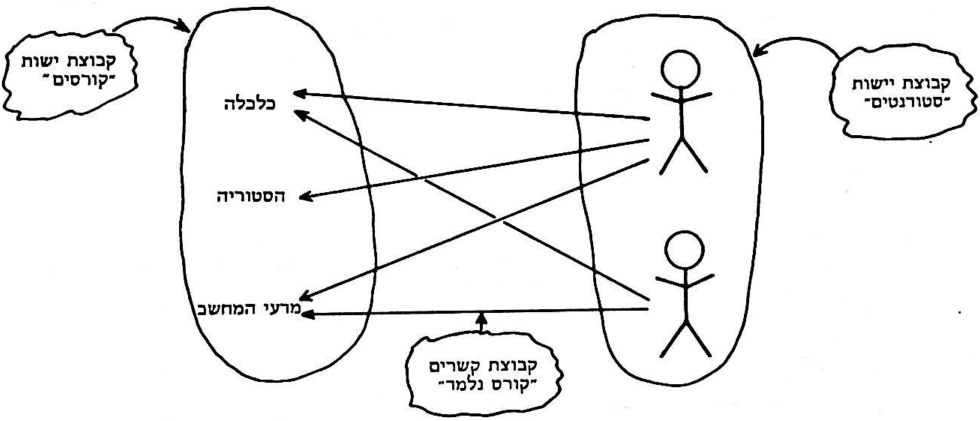
תרשים 17 – 3: ייצוג המציאות במודל ישויות - קשרים

תרשים 17—3 מתאר המרה של מרכיבי מציאות רלוונטית למודל של מערכת מידע המייצג את המציאות.

תכונה **קושרת** בדרך כלל את המאפיין לערך אחד מתוך תחום הערכים שלו ולכן היא מייצגת **עובדה** מהעולם האמיתי. בדרך כלל שם התכונה ושם תחום הערכים זהה. התכונה "שם הסטודנט" למשל, קושרת את המאפיין לערך מתוך התחום "שמות סטודנטיים". לא תמיד שם התכונה זהה לשם התחום כמו במקרה של התכונה "גיל" אשר יכולה לקבל את ערכיה מתוך תחום "מספר שנים".

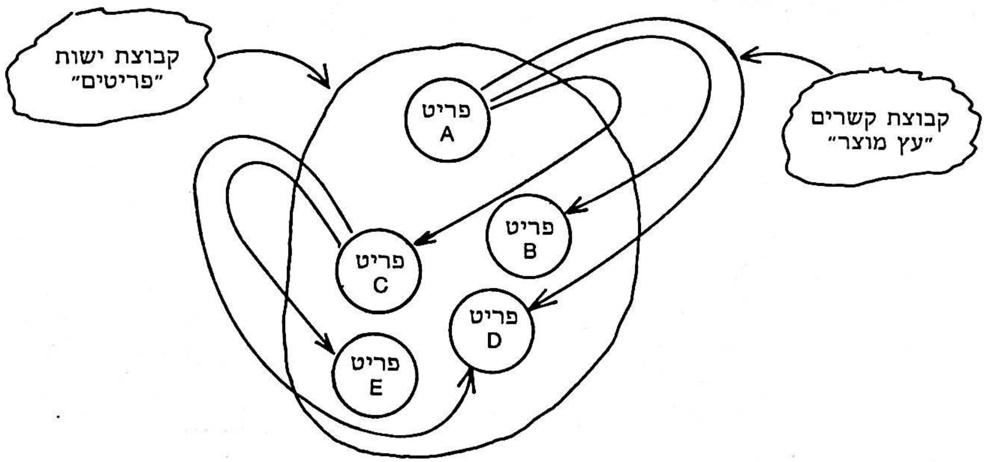
קבוצת קשרים (Relationship Set)

קבוצת קשרים הינה אוסף המופעים של הקשרים אשר מבטאים **יחס** כלשהו אשר קיים בין קבוצת ישות אחת לקבוצת ישות אחרת. קבוצת הקשרים הינה מבנה לוגי אשר מאפשר למנתח המערכות להתייחס אל המציאות מבלי שיהיה עליו להכיר את כל הקשרים הממשיים ולטפל בהם. קבוצת הקשרים "קורס נלמד" מגדירה את הקשר בין קבוצת הישות "סטודנטים" לקבוצת הישות "קורסים". קשר זה מציין איזה קורסים נלמדו על ידי איזה סטודנטים. ברגע שהוגדרה קבוצת הקשרים אפשר לטפל במבנה הלוגי הזה מבלי להכיר את הסטודנטים, הקורסים והקשרים ביניהם.



תרשים 18 – 3: קבוצת קשרים בין קבוצות ישות

קבוצת הקשרים יכולה להגדיר גם קשר בין ישויות שונות המשתייכות לאותה קבוצת ישות. "עץ המוצר" המתאר את מבנה המוצר הינו למעשה קבוצת קשרים בין מופע כלשהו של הישות "פריט" לבין מופעים אחרים של אותה ישות.



תרשים 19 - 3: קבוצת קשרים בין ישויות באותה קבוצת ישות

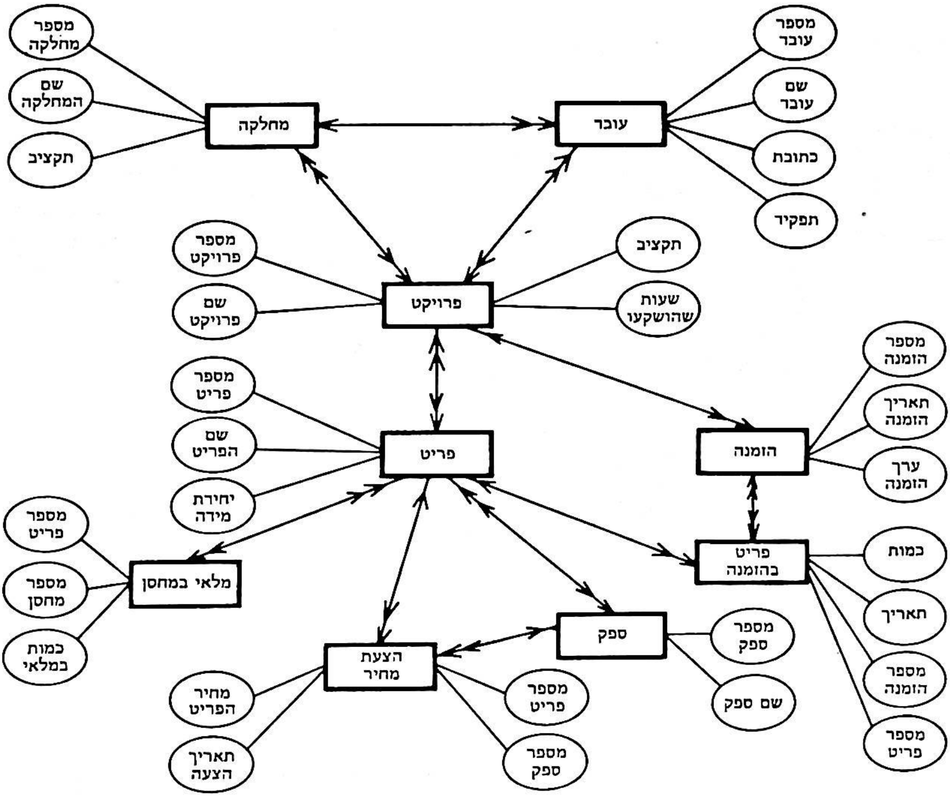
3.3 תרשימי ישויות־קשרים

נוח מאד להציג את מודל ישויות־קשרים באמצעות תרשים. בתרשים נשתמש בסימולים הבאים:

מלבן — קבוצת ישות,
עגול — תכונות של הישות,
חץ — קשר בין ישויות. סוג ההתאמה בין ישויות יסומן בחץ.

תרשים 20—3 מציג דוגמא של מודל ישויות־קשרים בחברה עסקית בתחום פיתוח מוצרים כלשהם. בתרשים מוצגות שמונה ישויות שונות עם המאפיינים שלהן. נבהיר חלק מהקשרים בין הישויות:

— הקשר פרויקט—עובד הינו רב—רב־ערכי כי מספר עובדים מועסקים בפרויקט אחד ועובד יכול לעסוק במספר פרויקטים במקביל.
— הקשר פרויקט—פריט הינו רב—רב־ערכי כי בפרויקט אחד יכולים להשתמש במספר פריטים שונים ופריט מסויים יכול להמצא בשימוש במספר פרויקטים שונים.
— הקשר פריט—הצעת־מחיר הינו חד—רב־ערכי כי לפריט אחד יתכנו מספר הצעות מחיר שונות, אבל הצעת מחיר מתייחסת לפריט אחד בלבד.



תרשים 20 – 3: תרשים ישויות־קשרים

4. מודל הנתונים

מודל הנתונים (Data Model) הינו מסגרת לוגית אשר מתארת את הישויות, התכונות והקשרים המייצגים את הנתונים בארגון. המודל נותן לכל המרכיבים שמות סימבוליים המאפשרים לטפל במבנים אלה בצורה לוגית. לתוך מסגרת זו יוצקים בהמשך את התוכן, כלומר את הערכים השונים שהתכונות מקבלות. שינוי בערכים אלה מייצג שינוי כלשהו שחל במציאות, אולם השינוי אינו משפיע על המודל אלא רק על תוכנו ברגע נתון. בהיותו מסגרת לוגית משמש המודל ככלי אינטלקטואלי המאפשר קיום תקשורת בין המשתמשים השונים בנתונים.

נשתמש בדוגמא כדי להסביר את מושג המודל. מפת הכבישים הינה למעשה **מודל לוגי** אשר מייצג את הקשר התחבורתי בין יישובים שונים. משרטטי המפה לא השתדלו לדייק בצורה מירבית בשרטוט כדי שכל עיקול או שיפוע בכביש יוצגו בה. המשמעות היא שמפת הכבישים איננה מבנה פיסי, כלומר תיאור מדויק של תוואי הכביש בין שתי נקודות, אלא **תיאור לוגי** של הכביש. בניית גשר או מסעף כלשהו הינם ללא ספק שינוי במבנה הפיסי אבל אינם מחייבים שינוי המפה מאחר והיא משמשת רק לייצוג לוגי של הכביש במודל. נוכל למצוא מפות מדוייקות המהוות תיאור פיסי של מערכת הכבישים כי יש להן שימושים נוספים, ולכן נדרשת בהן רמת דיוק רבה יותר של המודל.

בצורה דומה יש לבנות מסגרת לוגית ואוסף של כללים לייצוג המציאות במערכת המידע הממוחשבת. מודל הנתונים מאפשר את תרגום מודל ישויות—קשרים המייצג את המציאות למבנים לוגיים הניתנים לטיפול במערכות מחשב.

מקובל להבחין בין שלשה סוגי מודלים של נתונים במערכות DBMS:

א. **מודל תפישתי** (Conceptual Model): זהו המודל הבסיסי המייצג את הנתונים בארגון. המודל קובע את הישויות, התכונות והקשרים הרלוונטיים. הוא משמש לארגון הנתונים, לתכנון בסיס הנתונים וככלי להחלפת רעיונות בין המשתמשים השונים.

מודל הנתונים התפישתי מייצג בצורה הטובה ביותר את מבנה הנתונים הנכון עבור הארגון והוא בלתי תלוי במחשב או במערכת DBMS כלשהי.

ב. **מודל לוגי** (Logical Model): מודל הנתונים הלוגי הינו מודל נתונים המשמש את מערכת DBMS לייצוג המודל התפישתי. מקובל לכנות מודל זה **סכימה** (Schema) של המציאות מאחר והוא מייצג אותה במחשב. קיימים מספר רב של מודל לוגיים אולם רק שלשה מהם זכו לתפוצה נרחבת וגם מיושמים במערכת DBMS מסחריות: **המודל ההיררכי, המודל הרשתי והמודל הטבלאי**. קיימות מערכות אשר משתמשות בשילוב מסויים בין המודלים הבסיסיים האלה או שמשתמשות רק בחלק ממודל מסויים.

מאחר והמודל הלוגי משמש מערכת מחשב חלים עליו מגבלות מסויימות שנובעות מהצורך לטפל במִבני הנתונים בצורה יעילה ונוחה. הדרישה העיקרית להפעלת מערכת ניהול בסיסי נתונים היא ביצוע **מיפוי** בין המודל התפישתי לבין המודל הלוגי אשר כולל את כל האילוצים הדרושים. הפרקים הבאים יוקדשו להצגת שלשת המודלים הלוגיים הבסיסיים בצורה מעמיקה יותר.

ג. **המודל הפיסי** (Internal or Physical Model): את המודל הלוגי ניתן ליישם במערכת DBMS במספר מבנים פיסיים שונים. המשתמש מטפל במודל הלוגי ולכן אין לו גישה אל המודל הפיסי אשר הוא "שקוף" בעיניו. המודל הפיסי מתייחס לשיטת ייצוג המודל הלוגי

במחשב ולכן נמצא בו מושגים כגון יחידות אחסון, גושים, מאגרי קלט/פלט, אינדקסים, מצביעים וכדומה. **המיפוי** בין מודל הנתונים הלוגי לבין המודל הפיסי מתבצע על ידי מערכת DBMS.

בהמשך הספר נשתמש במושג **מודל נתונים** במובן הרחב. כלומר, הוא יכול להיות מודל תפישתי, מודל לוגי או מודל פיסי.

4.1 מרכיבי מודל הנתונים

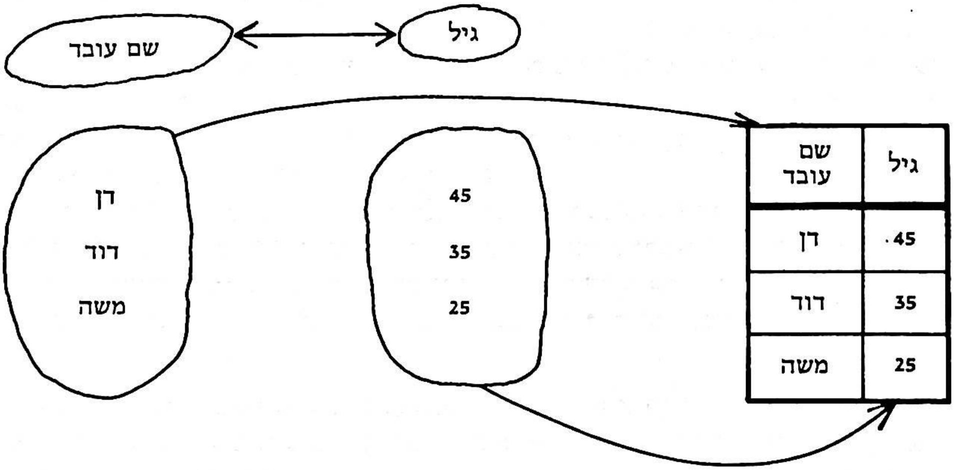
בסעיף הקודם הצגנו את האלמנטים אשר מתארים את המציאות במודל קשרים—ישויות. כדי שאפשר יהיה לעבד את המודל במחשב יש להסב אותו למבנה של מודל לוגי שבו מספר מרכיבים אשר יוסברו בהמשך.

שדה (Field, Data Item, Data Element)

השדה הוא יחידת הנתונים הבסיסית ביותר, או יחידה אטומית, במשמעות שאינו ניתן לחלוקה נוספת. השדה הינו האלמנט במודל הנתונים המשמש לייצוג **תכונה** של ישות והוא יכול לקבל **ערך** (Data Value) מתוך תחום הערכים של התכונה שאותה הוא מייצג. בצורה גרפית נסמן את השדה על ידי עגול, בדומה לתיאור תכונה בתרשימי ישויות־קשרים.

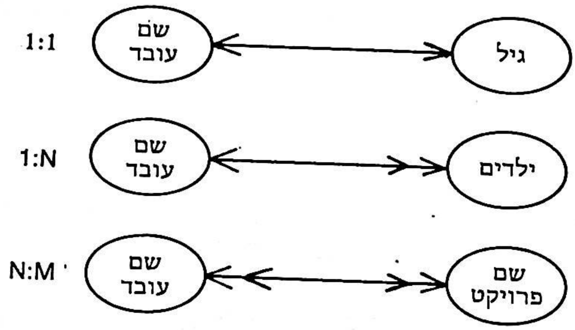
קשר בין שדות (Data Association, Relationship)

הקשר בין השדות מגדיר יחס כלשהו ביניהם, והוא אשר נותן להם משמעות. בצורה גרפית יסומן הקשר בחץ עם סימון בהתאם לסוג ההתאמה הקיימת בין השדות.
רישום נפרד של קבוצת הגילים בתרשים 3—21 הינו חסר משמעות מכיון שאין יודעים איזה גיל שייך לאיזה עובד. יצירת הקשר בין שני השדות, היא הנותנת את המשמעות "גיל העובד".



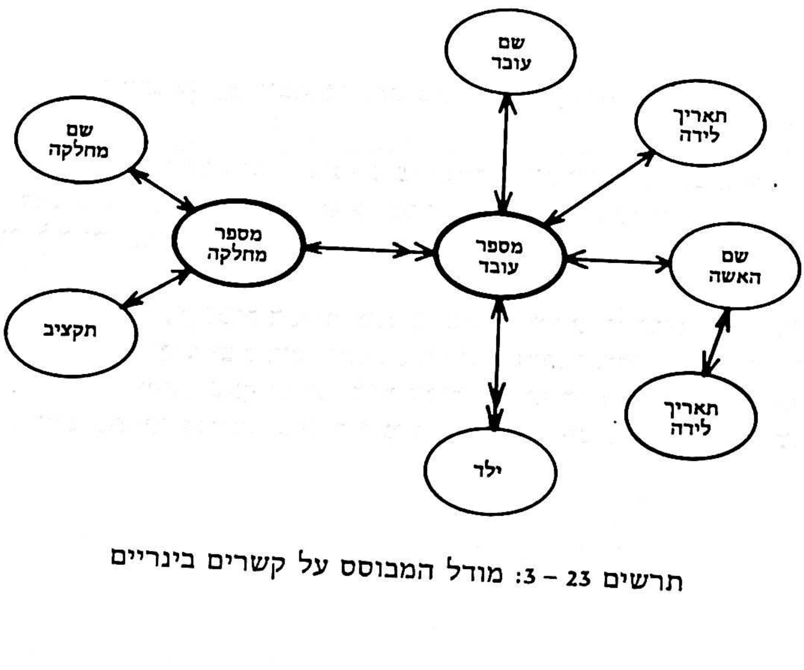
תרשים 21 - 3: קשר בין שני שדות

הקשר הבסיסי ביותר הינו הקשר הבינרי אשר קושר שני שדות. בין שני השדות יתכנו שלשת סוגי ההתאמה האפשריים, כפי שמוצג בתרשים 22—3.



תרשים 22 – 3: סוגי ההתאמות הקיימים בין שדות

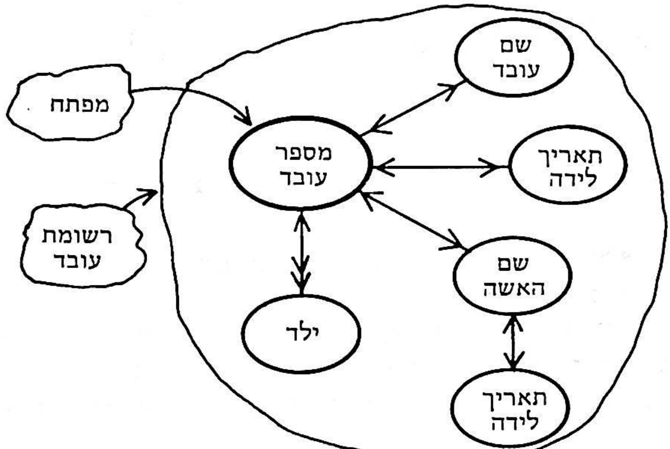
בין "שם עובד" ו"גיל" קיים קשר 1:1 כי לכל עובד יכול להיות גיל אחד בלבד. יש לשים לב לכך שהדיון הוא על התאמה בין שני שדות ולא בין הישות "שם עובד" והישות "גילים" שביניהן קיימת התאמה שונה לחלוטין. בין "שם עובד" ו־"ילדים" יש קשר 1:N כי לכל עובד יתכנו מספר ילדים, אבל לכל ילד יש רק אבא אחד. בין "שם עובד" ו־"שם פרויקט" יש קשר N:M כי עובד מסויים יכול לעבוד במקביל במספר פרויקטים ובפרויקט אחד מועסקים עובדים אחדים.



תרשים 23 – 3: מודל המבוסס על קשרים בינריים

רשומה (Record)

אוסף השדות המייצגים את התכונות של ישות אחת יקרא **רשומה**. ניתן לייצג את הישות על ידי אוסף של קשרים בינריים בין התכונות אך הוא אינו נוח לעיבוד במערכות מחשב כי הוא מחייב טיפול במספר רב של מרכיבים.



תרשים 24 – 3: ייצוג רשומה כאוסף של קשרים בינריים

במקום הייצוג הגרפי המשמש לייצוג רשומות במודל הנתונים ניתן להשתמש במבנה פשוט יותר — **הרשומה השטוחה** (Flat Record). במבנה זה כל השדות של הרשומה ימצאו בקירבה פיסית זה ליד זה, כמתואר בתרשים 25—3.

במודל הנתונים ירשמו שמות השדות ובבסיס הנתונים יופיעו הערכים שלהם, ולכן המבנה הפיסי של הרשומה אינו חייב להיות זהה למבנה הלוגי שלה. בבסיס הנתונים עשויים השדות להופיע בסדר שונה ויתכן שאף יצורפו להם שדות נוספים הדרושים לניהול בסיס הנתונים. בבסיס הנתונים מוצגים הנתונים במבנה לוגי כפי שהוא נתפס על ידי המשתמש.



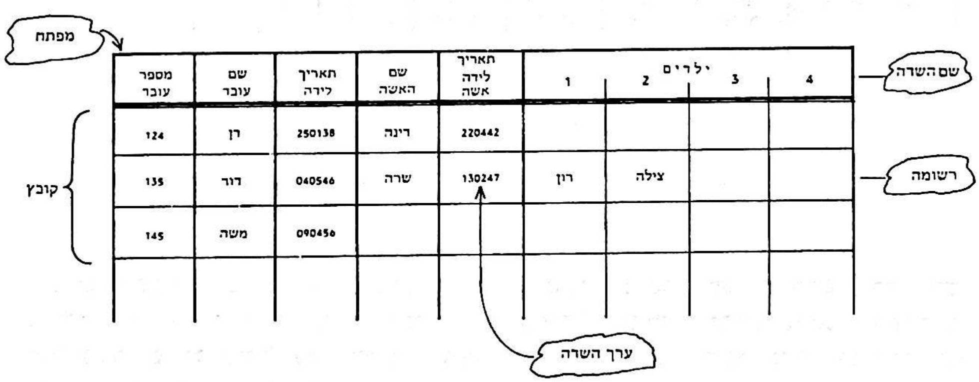
תרשים 25 – 3: רשומה שטוחה

הרשומה מייצגת ישות ולכן כל **מופע** (Occurrence) שלה בבסיס הנתונים **מייצג מופע של ישות במציאות.**

מבנה הרשומה השטוחה הוא המבנה המקובל ביותר לייצוג הקשרים בין התכונות של הישות. כפי שנראה בהמשך, כל המודלים של הנתונים משתמשים במבנה זה למטרה זו, כשם שמשתמשות בו גם המערכות לניהול קבצים.

סוג רשומה (Record Type)

אוסף הרשומות המייצגות ישות אחת יקרא **סוג רשומה.** לאוסף רשומות העובד נקרא ״סוג־רשומה עובד״ וכך, במסגרת סוג־רשומה אחד יתכנו מספר בלתי מוגבל של מופעים של רשומות. מונח מקובל אחר לסוג רשומה הוא **קובץ** (File). כל הרשומות מאותו סוג הן בעלות מבנה זהה, כלומר סדר הופעת השדות ברשומה הוא קבוע וזהה עבור כל המופעים. מן הראוי לציין שבקובץ אחד יכולות להיות רשומות מסוגים שונים.



תרשים 26 – 3: קובץ או סוג רשומה

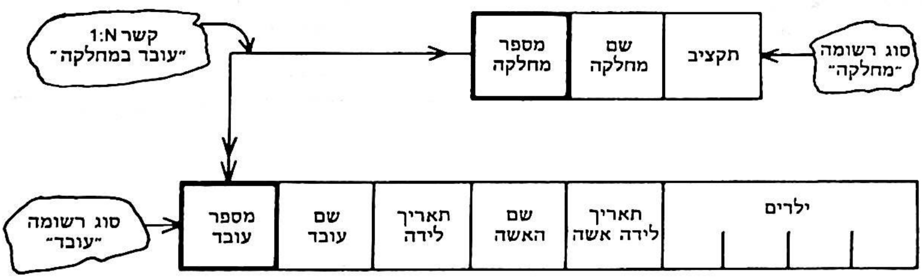
המפתח של הרשומה הוא השדה, או מספר שדות רצופים, אשר מזהים אותה בצורה חד משמעית. מערכת DBMS משתמשת בתוכן שדה המפתח כדי לקבוע את הכתובת הפיסית בבסיס הנתונים וכאמצעי לאיתור הרשומה. רוב מערכות DBMS מאפשרות הגדרת מפתחות משניים לכל קובץ כדי לאפשר איתור מהיר של רשומות לפי מפתח אשר שונה מהמפתח הראשי שלהן.

חשוב להדגיש שאין המבנה הפיסי של הקובץ חייב להיות זהה למבנה הלוגי. בתוך בסיס הנתונים הרשומות יכולות להופיע בסדר שונה ומבנה הרשומה יכול להיות שונה כי זהו מבנה לוגי כפי שהוא מובן על ידי המשתמש.

קשרים בין סוגי רשומות (Record Association, Relationship)

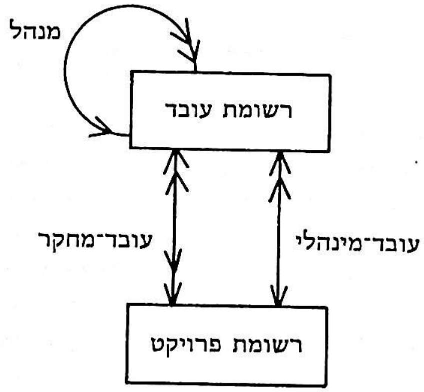
הקשר בין סוגי רשומות שונים מגדיר יחס כלשהו ביניהן והוא מייצג את הקשרים בין קבוצות הישות השונות. בדומה לקשרים הקיימים בין שדות, גם הקשרים בין סוגי רשומות יכולים להיות בכל אחד משלשת סוגי ההתאמה האפשריים.

בתרשים 27—3 מוצג קשר חד—רב־ערכי בין שני סוגי רשומה שונים. בין שני סוגי רשומה יתכנו מספר קשרים שונים וכן יתכן קשר בין רשומות שונות השייכות לאותו סוג רשומה.

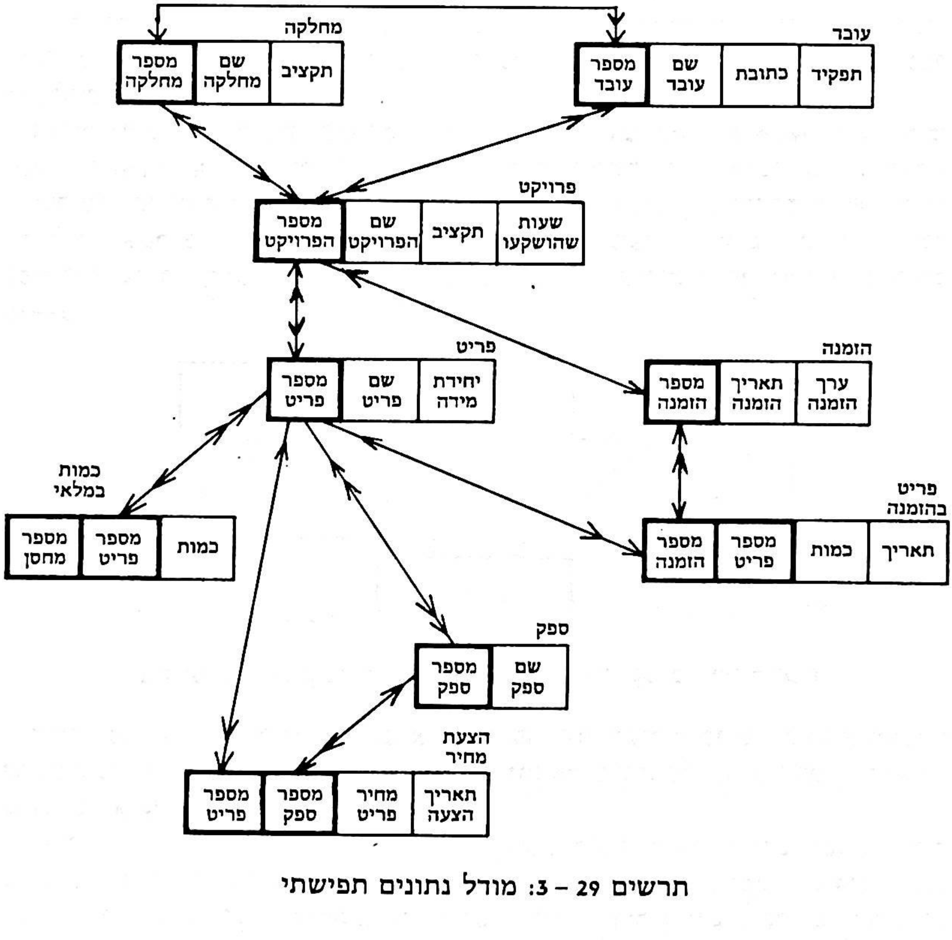


תרשים 27 - 3: קשר חד - רב־ערכי בין שני סוגי רשומות

בדוגמא שבתרשים 28—3 מודגמים קשרים שונים בין שני סוגי רשומות. הקשר החד—רב־ערכי "מנהל" מציג את הקשר בין עובד המוגדר כמנהל לבין כל העובדים שהוא אחראי להם. לכל עובד יש רק מנהל אחד. הקשר "עובד־מחקר" מציג את הקשר הרב—רב־ערכי בין רשומות פרויקט לרשומות עובדים שהם עובדי מחקר. הקשר "עובד־מינהל" מציג את הקשר הרב—חד־ערכי בין רשומות פרויקט לרשומות עובדים שהם עובדים מנהליים. מבנה הקשרים מציין את העובדה שעובד מנהלי יכול לעבוד רק בפרויקט אחד, בשעה שעובד מחקר יכול להשתתף במספר פרויקטים במקביל.



תרשים 28 - 3: קשרים שונים בין סוגי רשומה



תרשים 29 - 3: מודל נתונים תפישתי

בתרשים 29—3 ניתן תיאור של מודל נתונים אשר מבוסס על תרשים ישויות—קשרים שהופיע בתרשים 20—3.

5. המודלים הלוגיים העיקריים

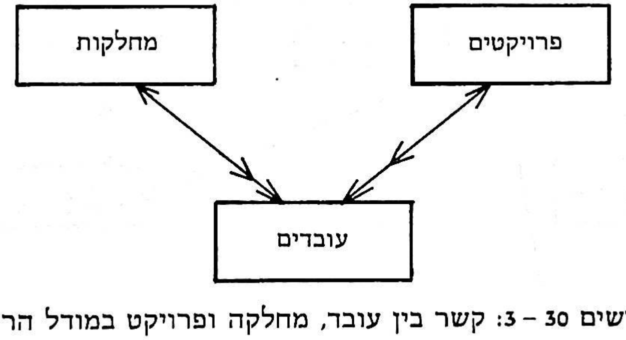
בסעיף זה נדון בשלשת המודלים הלוגיים החשובים והנפוצים ביותר אשר מיושמים ברוב מערכות DBMS הקיימות בשוק. אפשר לתאר את המציאות בעזרת כל אחד מהמודלים במידה שונה של פשטות, בהירות ונוחות. על כן נבין את הקושי שבבחירת מערכת DBMS אשר תשרת בהצלחה את רוב מערכות עיבוד הנתונים בארגון. כל המודלים מייצגים את הקשרים בין התכונות של הישות באמצעות הרשומה, אך הם נבדלים זה מזה בעיקר בצורה של הצגת הקשרים בין הישויות ובהצגת הקשרים בין הרשומות.

בסעיף זה ינתן תיאור תמציתי של המודלים ובהמשך יוקדש פרק נפרד לכל אחד מהם.

5.1 מודל הרשת (Network Data Model)

מודל הרשת הוגדר בצורה פורמלית על ידי צוות עבודה בנושא בסיסי נתונים בתחילת שנות ה־70. קיימות מערכות שונות אשר מבוססות על מודל הרשת אבל הן אינן עומדות בכל ההמלצות שלו.

המודל מאופיין על ידי הגדרת שני סוגי רשומות: **רשומת אב** (Owner Record) **ורשומת בן** (Member Record). בין שני סוגי רשומות אלה קיים קשר חד־רב־ערכי (1:N). לדוגמא נסתכל על קבוצות הישות "מחלקות", "פרויקטים", "עובדים", קיים ביניהם הקשר "עובד במחלקה" אשר מגדיר התאמה מקבוצת "מחלקות" לקבוצת "עובדים", והקשר "עובד בפרויקט" שמגדיר התאמה מקבוצת "פרויקטים" לקבוצת "עובדים", כפי שמתואר בתרשים 30—3.



תרשים 30 - 3: קשר בין עובד, מחלקה ופרויקט במודל הרשת

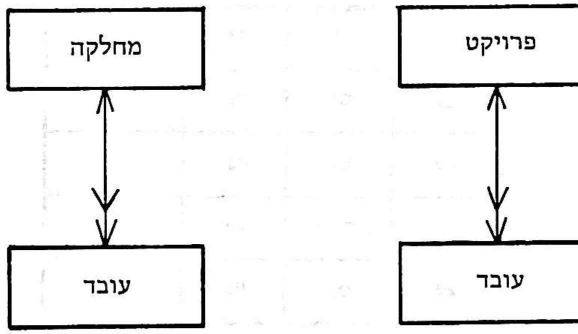
המודל מאפשר לקשור רשומת אב אחת למספר סוגי רשומות בן שונות והוא מאפשר לרשומת בן אחת להיות קשורה למספר סוגי רשומות אב שונות. כל קשר ביניהם יהיה תמיד 1:N, קשר N:M יוסב לשני קשרים מסוג 1:N.

מודל הרשת מיושם במספר רב של מערכות DBMS מסחריות והוא שימש נושא לעבודות מחקר שונות. המודל מאפשר גישה יעילה יחסית לנתונים, אך דורש מהמשתמש להגדיר את המבנים הפיסיים של הנתונים ואת מסלולי הגישה אליהם. חוסר הגמישות של המודל מתבטא בעובדה שהוספת מסלולי גישה חדשים מחייבת טעינה מחדש של בסיס הנתונים. המבנים הפיסיים שבהם משתמשים במצביעים נוטים להיות מורכבים למדי.

5.2 מודל הירככי (Hierarchical Data Model)

המודל ההירככי הינו ותיק יותר ממודל הרשת ונמצא בשימוש זמן רב במערכות DBMS רבות. גם במודל ההירככי משתמשים בשני סוגי רשומות, **רשומת אב** ו**רשומת בן** שביניהן קיים קשר חד—רב־ערכי (1:N). ההבדל העיקרי בין מודל זה למודל הרשת נובע מהעובדה שרשומת בן יכולה להיות קשורה לרשומת אב אחת בלבד. המבנה המתקבל כתוצאה מצורת קשר זו הוא מבנה של **עץ נתונים**, כאשר לכל רשומה בעץ יש מסלול גישה אחד בלבד.

נתבונן בקשר בין קבוצות הישויות "מחלקות", "עובדים", "פרויקטים". העובד קשור לרשומת המחלקה שלו וקשור לרשומת הפרויקט שבו הוא מועסק. מכיון שהמודל ההירככי אינו מאפשר לרשומת הבן "עובד" להיות קשורה לשתי רשומות אב שונות, דרושים שני מבנים הירככים נפרדים כמתואר בתרשים 31—3. שילוב שני המבנים ההירככיים האלה נעשה על ידי קשר לוגי נוסף המאפשר גישור על פני מבנים הירככיים שונים. מגבלה זו גורמת לבניית מודל נתונים מורכב יותר ממודל הרשת מבחינת ייצוג הקשרים מאחר ובמודל הרשת מיוצגים הקשרים בצורה טבעית יותר. אך לעומת זאת מכיון שלכל רשומה יש מסלול גישה יחיד, ניתן ליישם את המודל בצורה פשוטה יותר.

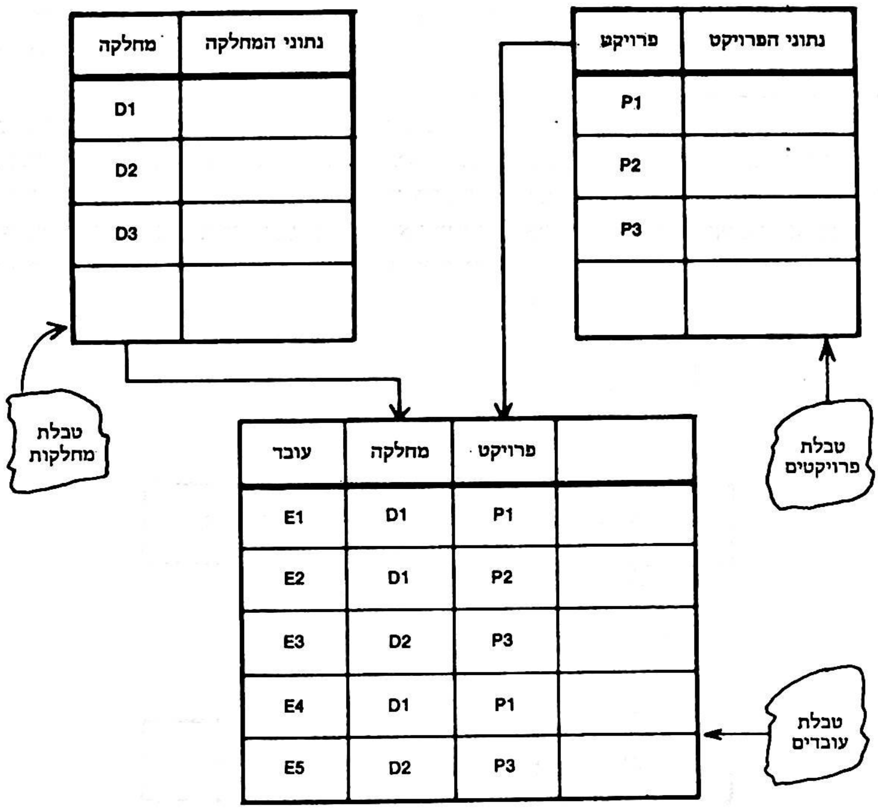


תרשים 31 - 3: קשר בין עובד, מחלקה ופרויקט במודל הירככי

5.3 מודל טבלאי (Relational Data Model)

המודל הטבלאי פותח על ידי E.F. Codd והוצג על ידו בראשית שנות ה־70, ומאז פותחו מערכות DBMS אחדות אשר מבוססות על מודל זה. המודל הטבלאי מאפשר את תיאור הנתונים בצורה טבעית ביותר, ללא הוספת מבנים מיוחדים ושדות אשר מטרתם העיקרית היא סיוע לייצוג הנתונים במחשב, ובכך הוא מאפשר אי־תלות נתונים מירבית. המודל הטבלאי

מציג את כל הנתונים באמצעות טבלאות, כאשר כל טבלה מתארת **יחס** (Relation) המגדיר קשרים בין מאפיינים. ייצוג הקשרים בין ישויות מתאפשר באמצעות מאפיינים זהים בשתי טבלאות שונות. בדוגמא שבתרשים 32—3 נוצר הקשר בין הטבלאות באמצעות המאפיינים "מחלקה" ו"פרויקט".



תרשים 32 – 3: קשר בין עובד, מחלקה ופרויקט במודל טבלאי

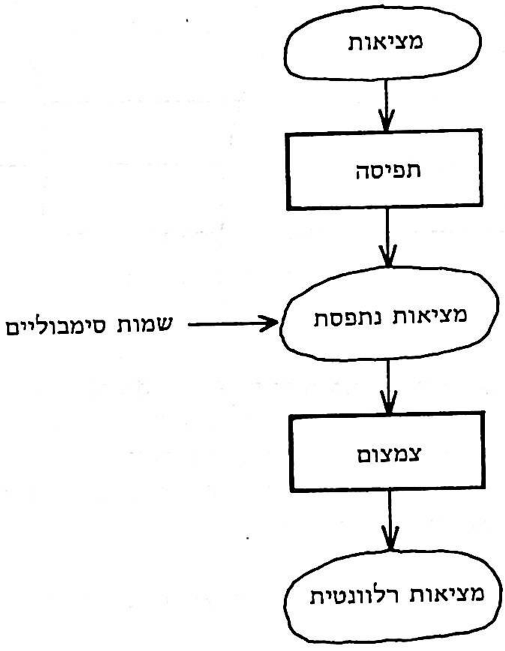
המודל הטבלאי זוכה היום להתעניינות רבה ולכן גם למחקר מעמיק, בעיקר בגלל תכונותיו המעניינות, השירותים שהוא מציע ואי־תלות בנתונים שהוא מאפשר. הגמישות שלו במתן שירותי אחזור נתונים גורמת ליעילות גבוהה של המשתמש בו למטרת שאילתא, אך יעילותו נמוכה, בהשוואה למודלים אחרים, בניצול זמן המחשב. יש לצפות שהמחקרים יביאו לאופטימיזציה בתחום זה ויתרמו, בנוסף לירידה בעלות החומרה והשיפורים בביצועי המחשב, לפיתוח מערכות DBMS נוספות אשר תתבססנה על המודל הזה.

6. סיכום

בפרק זה הצגנו את המושגים העיקריים הקשורים לתהליך בניית מודל הנתונים ששלביו המרכזיים הם:

א. הפיכת המציאות למציאות רלוונטית

השלב הראשון הוא הבנת והגדרת המציאות הרלוונטית של הארגון והנתונים הרלוונטיים שיש לנהל במערכת המידע. בשלב הזה מזהים את המציאות הרלוונטית של הארגון, מגדירים את המודל וקובעים את האלמנטים שלו ונותנים להם שמות סימבוליים. **המציאות הנתפשת** היא רחבה מדי ואין אפשרות ועניין לטפל בה בתוך המודל, ולכן יש צורך לזהות את **המציאות הרלוונטית** אשר משמשת בסיס למודל.



תרשים 33 – 3: הפיכת המציאות למציאות רלוונטית

ב. הקבצת האלמנטים השונים

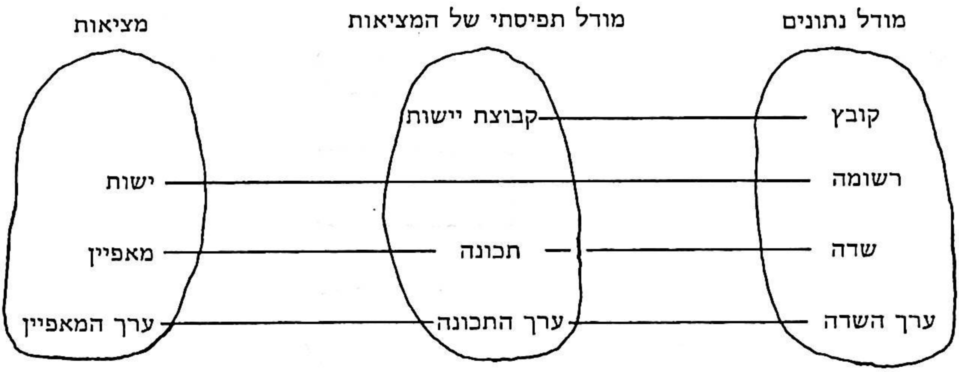
לבניית מערכת המידע דרושה הקבצת היישויות והקשרים לשתי קבוצות הומוגניות. כפי שכבר הודגש, אחת הבעיות בשלב זה היא קביעת צורת הייצוג של האלמנטים, כמאפיינים של ישות או כישות עצמאית. בשלב הזה מזהים את כל הקשרים הקיימים בין הנתונים ומסווגים אותם.

ג. בניית המודל התפיסתי של הנתונים

המודל התפישתי מתאר את כל מבני הנתונים הדרושים למערכת המידע, והוא צריך להיות חופשי מאילוצים הנובעים משימוש במחשב. תוך כדי בניית המודל מנתח המערכות אינו צריך להתייחס לצורת היישום של המודל במערכת אמיתית כלשהי של ניהול מערכות בסיסי נתונים.

ד. בניית מודל הנתונים

בשלב זה יש לתרגם את המודל התפישתי לאחד מבין המודלים הקיימים של מערכות בסיסי נתונים. אם קיימת כבר בארגון מערכת DBMS, יש לתרגם את המודל התפישתי למודל הנתונים המהווה בסיס למערכת הקיימת.



תרשים 34 – 3: תהליך בניית מודל הנתונים

7. שאלות

1. נניח שנתונות הקבוצות הבאות:
   — קבוצת הסטודנטים [S_1, S_2.....Sn]
   — קבוצת המרצים [P_1, P_2....Pm]

   הגדר את הקשרים הבאים ותן מספר דוגמאות ע״י הצגת זוגות מסודרות:
   — מלמד את...
   — מקבלים יעוץ על ידי... (הנח שכל סטודנט מקבל יעוץ ע״י מרצה אחד בלבד)
   — נבחן על ידי...
2. נתח את הקטע הבא וזהה ישויות, תכונות של ישויות, ערכים וקשרים. תן דוגמאות. ״מספר הרשוי של המכונית 456־123. המכונית נשכרה לשבוע החל משעה 7.30 בתאריך 1/8/83 מחברת רכב־סע״.
3. הגדר מספר תכונות אלטרנטיביות שיכולות לשמש כמפתחות ראשיים הגדר את התחום של כל אחת מהן. הסבר את התכונות והחסרונות של כל תכונה המשמשת כמפתח במערכת ממוחשבת המטפלת ב:
   — ניהול כוח אדם במפעל.
   — הזמנת כרטיסי טיסה.
   — רישום סטודנטים באוניברסיטה.
   — רישום בעלות על מקרקעין.
4. ניתן לבנות קשר מפורש בין שתי ישויות ע״י שימוש בתכונה. לדוגמא, בקשר בין ״בית״ ל״אדם״ נשתמש בשם בעל־הבית ברשומת הבית. הסבר מה ההבדל בין סוג קשר כזה לבין יישום הקשר באמצעות מצביעים למשל.
5. הגדר את סוגי הקשרים המופיעים במשפטים שבקטע הבא:
   האוניות עוגנות בנמלים. לכל אניה יש תאריך הגעה ותאריך הפלגה. ניתן לטעון כל אוניה במספר מכולות. כל מכולה נמצאת במעבר מנמל מקור לנמל יעד כלשהו. מספר מכולות מהוות משלוח אחד. יתכן גם שמשלוח מכיל מכולה בודדת.
6. שאלה זו בוחנת את יכולתך לזהות קשרים וישויות. הנך מתבקש לנתח את המשפטים הבאים, אשר עוסקים באוניברסיטה כלשהי:
   — מרצה יכול ללמד במספר קורסים.
   — מרצה יכול להיות יועץ של מספר סטודנטים או של שום סטודנט. לכל סטודנט יש יועץ אחד לכל היותר.
   — לכל מרצה יש חדר בקמפוס, אך יתכן שלמרצה אחד יהיה יותר מחד אחד או ששני מרצים ישתמשו באותו חדר.
   — סטודנט רשום למספר קורסים.
   — קורס נלמד תמיד בכיתה אחת. מספר קורסים שונים יכולים להיות מועברים באותה כיתה.
   — עבור קורס מסויים קיימים מספר קורסים אחרים המהווים תנאי מוקדם.

   א. תאר מספר תכונות של הישויות שזיהית. זהה את הקשרים בין הישויות.
   ב. צייר דיאגרמת ישויות־קשרים.

7. זהה את הישויות, הקשרים והתכונות במשפטים שבקטע הבא:
קיימות מספר חברות תעופה. כל אחת מהן מבצעת טיסות בין נמלי תעופה מסויימים. נמלי התעופה מזוהים ע"י קוד בן שלשה תווים. הטיסות ממוספרות, לדוגמא טיסה BA103 ממריאה בשעה מסויימת ונוחתת בשעה מסויימת. נוסע יכול להרשם לנסיעה אשר מורכבת ממספר טיסות בתאריכים שונים ובאמצעות חברות תעופה שונות.

פרק ד

מערכות לניהול בסיסי נתונים — מרכיבים ועקרונות פעולה

1. מבוא
2. מודל כללי של מערכת DBMS
3. הסכימה
4. תת־סכימה
5. סכימה פנימית
6. שפה לטיפול בנתונים
7. עקרונות הפעולה של מערכת DBMS
8. מילון נתונים
9. שפת שאילתות
10. שאלות

1. מבוא

בפרק זה נתאר את המרכיבים של מערכות DBMS, את עקרונות פעולתן והשירותים שמערכות אלה מעמידות לרשות המשתמשים. נציין מיד שלא כל המערכות המסחריות הקיימות ופועלות היום מכילות את כל המרכיבים שיפורטו. המבנה הכללי של מערכות DBMS שבו נדון אינו המבנה היחידי האפשרי, אך השתדלנו להציג באמצעותו את המבנים העיקריים והנפוצים ביותר.

מערכות DBMS התפתחו מתוך המערכות לניהול קבצים ונועדו להתגבר על רוב הבעיות הקיימות במערכות אלה. המערכות לניהול קבצים סיפקו כלים לאחסון ולשליפת רשומות מתוך קבצים. השליפה היתה של רשומה בודדת ובדרך כלל לפי מפתח אחד. במערכות DBMS מיושם עקרון אי־התלות בנתונים אשר מטרתו לבודד את תוכניות היישום מהמבנים הפיסיים של הנתונים ולאפשר להן טיפול במבנים לוגיים בלבד.

במערכות לניהול קבצים המשולבות עם שפת COBOL למשל, התיאור וההגדרה של הנתונים נעשים בתוך כל תוכנית יישום בקטע תוכנית נפרד אשר מיועד למטרה זו: Data Division. כל תוכנית יישום מתארת ומגדירה לעצמה את מבנה הנתונים שבהם היא מטפלת.

מערכות DBMS מאפשרות תיאור והגדרה של הנתונים באופן חיצוני לתוכניות היישום, וזהו הבסיס לאי־תלות בנתונים. בנוסף לשירותי הגדרת נתונים, מספקות מערכות DBMS שירותי גישה וטיפול בנתונים המאוחסנים בבסיס הנתונים. תוכניות היישום פונות לבסיס הנתונים רק באמצעות שירותים אלה ובכך מטפלות במבנים לוגיים בלבד.

את מערכות DBMS אפשר לסווג בשתי קבוצות עיקריות:

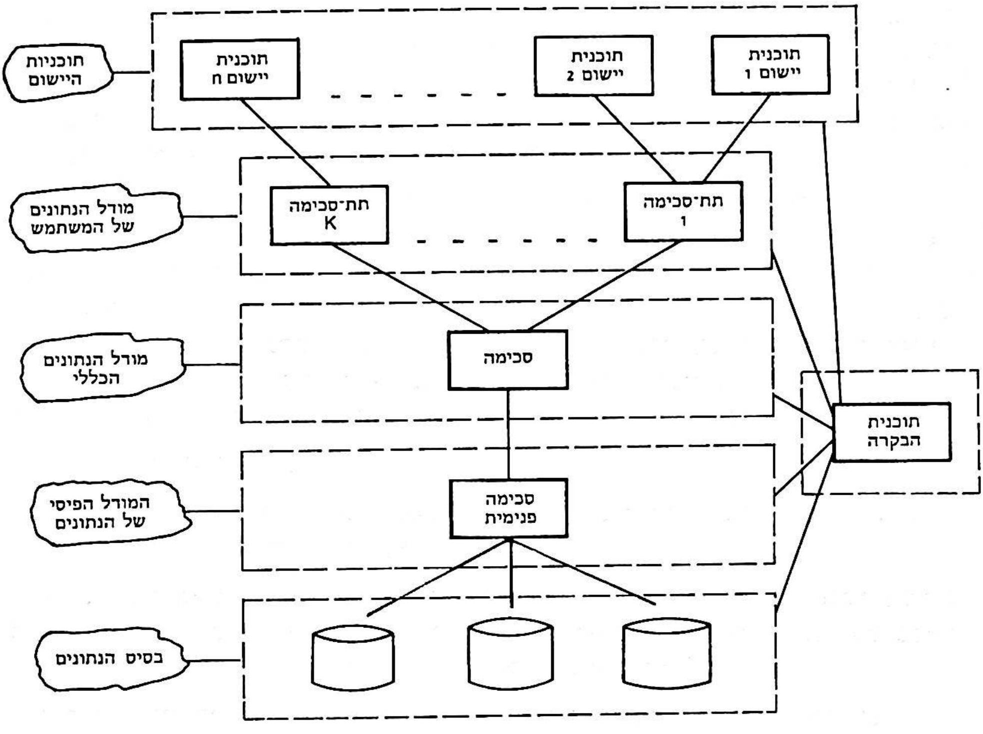
— **מערכות עצמאיות** (Self Contained Systems), בהן קיימת שפה עילית המאפשרת הגדרה, אחסון, שליפה וטיפול בנתונים.

— **מערכות המשתמשות בשפה עילית מארחת** (Host Based Systems) אשר עוסקות בעיקר בהגדרת הנתונים, אחסון ושליפה. הטיפול בנתונים ברמת היישום מתבצע בשפה המארחת, COBOL או PL1 למשל, באמצעות הפקודות הרגילות שלה.

בהמשך נעסוק במערכות השייכות לקבוצה השניה ונציג את מרכיביהן העיקריים.

2. מודל כללי של מערכת ניהול בסיסי נתונים

המודל הכללי של מערכת ניהול בסיסי נתונים המוצג בהמשך מבוסס על המלצות ועדת התקינה ANSI/X3/SPARC.



תרשים 1 – 4: מודל כללי של מערכת DBMS

תרשים 1—4 מציג את המרכיבים העיקריים של מערכת DBMS. כפי שניתן לראות בנוייה מערכת DBMS משישה רבדים עיקריים:

א. **תוכניות היישום** (Application Programs) אשר מעבדות את הנתונים בהתאם לדרישות הפונקציוליות של מערכת המידע. תוכניות היישום הן המודל הפונקציונלי של המציאות, כלומר הן מייצגות את האירועים וזרימת הנתונים בארגון. שחרור חולה מבית החולים למשל, יבוא לידי ביטוי במערכת המידע באמצעות תוכנית יישום שתעדכן את בסיס הנתונים בהתאם לאירוע.

תוכניות היישום מטפלות במבנים לוגיים המופיעים בתת־הסכימה המתאימה באמצעות שפת המחשב המארחת, למשל COBOL, אך לשם גישה לנתונים המאוחסנים בבסיס הנתונים הן משתמשות בפקודות מיוחדות של מערכת DBMS אשר משולבות בתוך תוכנית היישום. באמצעות פקודות אלה תוכנית היישום מאחסנת, שולפת ומעדכנת את הנתונים.

ב. **מודל הנתונים של המשתמש** מתאר את נקודת המבט של המשתמש ומגדיר את המבנה הלוגי המטופל בתוכניות היישום,

ג. **מודל הנתונים הכללי** מכיל את סכימת תיאור הנתונים,

ד. **המודל הפיסי של הנתונים** מוצג בסכימה הפנימית,

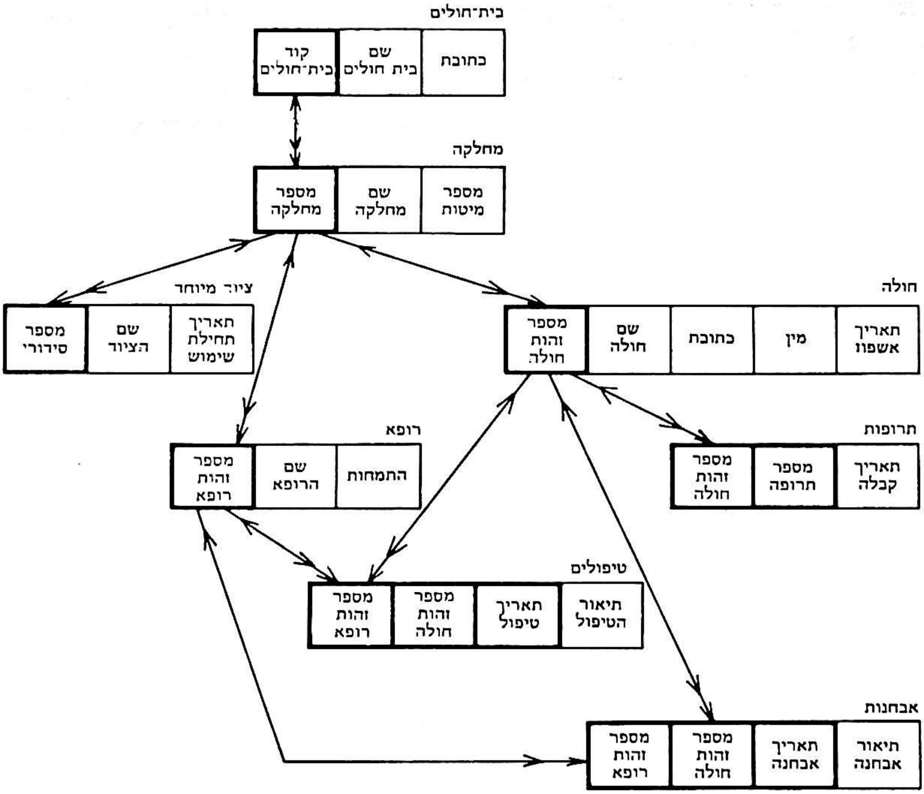
ה. **בסיס הנתונים** מאחסן את הנתונים,

ו. **תוכנית הבקרה** (Data Base Control System) מפקחת על תזרים העבודה במערכת ניהול בסיס הנתונים. תוכנית הבקרה מקבלת את הבקשה מתוכנית היישום, מבצעת את המיפוי לצורך קביעת מסלול שליפת הנתונים ולבסוף, שולפת את הנתונים ומעבירה אותם לתוכנית היישום. בדרך כלל משתמשת תוכנית הבקרה בשירותי מערכת ההפעלה לביצוע הגישה הפיסית לנתונים שביחידות האחסון. בנוסף למטלות אלה מטפלת תוכנית הבקרה במשימות נוספות כמו אבטחת נתונים, גיבוי והתאוששות.

3. הסכימה

הסכימה (Global View, Conceptual Schema, Schema) היא **תיאור פורמלי מפורט של מודל הנתונים הכללי**, כלומר של כל הנתונים אשר יופיעו בבסיס הנתונים. הסכימה מתארת את כל היחידות הלוגיות הקיימות במודל הנתונים (כגון רשומות, שדות, קשרים) ואת התכונות שלהן (סוג הרשומה, סוג השדה, אורך השדה, סוג הקשר). כל היחידות הללו מקבלות שמות סימבוליים כדי לפשט ולהקל על הפנייה אליהן.

הסכימה הינה אחד המאפיינים העיקריים של מערכות DBMS. היא מאפשרת הגדרת נתונים באופן חיצוני לתוכניות היישום, ומשמשת כאמצעי קשר בין תוכניות היישום לבין בסיס הנתונים.



תרשים 2 – 4: סכימה של מערכת מידע רפואי

כל מערכות DBMS מספקות **שפה להגדרת הסכימה** (DDL-Data Definition Language). שפה זו הינה שפה עילית אשר מאפשרת:

— לתת שם סימבולי לסכימה ולכל היחידות הלוגיות המתוארות בה כגון רשומות ושדות.

— לתאר את הקשרים בין היחידות הלוגיות ולתת שם סימבולי לקשר,
— להגדיר את המפתחות של הרשומה,
— לקבוע את תחום הערכים האפשריים ואת האילוצים על תחום זה,
— לקבוע את כללי ההגנה על הנתונים שבבסיס הנתונים.

דוגמא לשפה להגדרת סכימה על פי המלצות CODASYL ניתנת להלן. היא מתארת שתי רשומות ואת הקשר ביניהן על פי תרשים 2—4. הסבר מפורט של המונחים השונים בשפה ינתן בפרקים הבאים.

```
                 SCHEMA NAME IS HOSPITAL,  ←———  שם הסכימה
                   PRIVACY LOCK IS ABCD,  ←——  סיסמא
                   RECORD NAME IS PHYSICIAN,  ←———  שם רשומת הרופא
                     LOCATION MODE IS CALC USING PHYSICIAN-ID.  ←——  מפתח
תיאור שדות    {  01   PHYSICIAN-ID           PICTURE IS 9(5).
ברשומת רופא   {  01   NAME                   PICTURE IS X(30).
              {  01   SPECIALIZATION         PICTURE IS X(40).

                   RECORD NAME IS TREATMENT,  ←———  שם רשומת הטיפול
                     LOCATION MODE IS CALC USING TRT-ID.  ←———  מפתח
              {  01   TRT-ID.
תיאור שדות    {       03  PHYSICIAN-ID       PICTURE IS 9(5).
ברשומת טיפול  {       03  PATIENT-ID         PICTURE IS 9(8).
              {       03  DATE               PICTURE IS 9(6).
              {  01   DESCRIPTION            PICTURE IS X50).

שם הקשר בין   { SET NAME IS TREATMENT-GIVEN-BY-PHYSICIAN,
רופא לטיפול   {   MODE IS CHAIN LINKED PRIOR,  ←———  סדר הרשומות בקשר
                  ORDER IS SORTED,
שם רשומת האב ——→  OWNER IS PHYSICIAN,
שם רשומת הבן ——→  MEMBER IS TREATMENT LINKED TO OWNER,
                     ASCENDING KEY IS TRT-ID.
```

הגדרת הסכימה כוללת בעיקר את תיאור המבנה והקשרים שבין הנתונים אך קיימות מערכות DBMS אשר מאפשרות גם להגדיר או לקבוע בסכימה דברים נוספים:

תחום ערכים: ניתן להגדיר בסכימה את תחום הערכים החוקי שנתון כלשהו יכול לקבל. הגדרת תחום זה בסכימה משחררת את תוכניות היישום מבדיקה זו כי מערכת DBMS תבצע אותה באופן אוטומטי לפני כתיבת הרשומה בבסיס הנתונים לאחר קלט למשל. אם בדיקות התחום מורכבות יותר ניתן להגדיר **שגרה** (Data Base Procedure) מיוחדת שתמצא בספריה המנוהלת על ידי מערכת DBMS. לפני כתיבת רשומה חדשה לבסיס הנתונים, תפעיל תוכנית הבקרה את השגרה לביצוע הבדיקה. שגרה זו יכולה לבצע בדיקות מורכבות מאד כלשהן, כמקובל בכל תוכנית.

בדיקת תחום

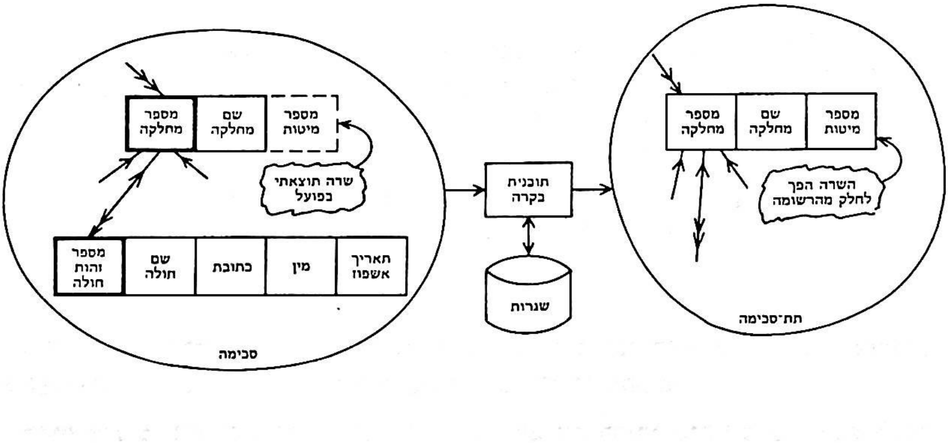
```
RECORD NAME IS DEPARTMENT
LOCATION MODE IS CALC USING DEP-ID.
01   DEP-ID          PICTURE IS 9(2) CHECK VALUE 01 THRU 62.
01   DEP-NAME        PICTURE IS (30).
01   NUMB-BED        PICTURE IS 9(3) CHECK IS NUMB-BED-CHECK.
```

שגרת בדיקה

שדה תוצאתי ממשי (Actual Result): ניתן להגדיר בסכימה שדה אשר יופיע בבסיס הנתונים בתוך הרשומה אבל את ערכו הוא יקבל משגרה מיוחדת ולא מתוכנית היישום.

```
01   FIELD-A PICTURE IS 9(5)
     IS ACTUAL RESULT OF DB-PROC-105.
```

שדה תוצאתי בפועל (Virtual Result): ניתן להגדיר בסכימה שדה אשר אינו מופיע בבסיס הנתונים מכיון שהוא שדה תוצאתי שערכו נקבע על סמך תהליך חישוב כלשהו המוגדר בשגרה מיוחדת. שדה זה יועבר לתוכנית היישום כאילו היה שדה ממשי. לדוגמא, במקום להחזיק ברשומת המחלקה את מספר המיטות במחלקה ניתן להגדיר שדה זה כשדה תוצאתי בפועל. כאשר נדרש נתון "מספר מיטות" תפעיל תוכנית הבקרה שגרה אשר תסרוק את כל רשומות החולים השייכים למחלקה, תחשב את מספר המיטות ותמסור את התוצאה בשדה המתאים.



תרשים 3 – 4: שדה תוצאתי בפועל

שדה מקור ממשי (Actual Source): ניתן להגדיר בסכימה שדה אשר ערכו יהיה זהה לערכו של שדה אחר אשר מופיע בפועל בבסיס בנתונים. בעת הפנייה לשדה מתוך תוכנית היישום תקבע תוכנית הבקרה את ערכו ותמסור אותו כנדרש. לדוגמא, אפשר להגדיר שדה מקור ממשי

"מספר מחלקה" בתוך רשומת החולה. לפני שרשומת החולה תוצג לתוכנית היישום תעביר תוכנית הבקרה את תוכן השדה "מספר מחלקה" מרשומת המחלקה אל השדה "מספר מחלקה" שברשומת החולה. לדוגמא,

שדה ברשומת חולה

```
01   DEP-NUMB PICTURE IS 9(2)
       IS ACTUAL AND SOURCE IS DEP-ID
       OF OWNER OF DEP-PATIENT.
```

שדה המקור ברשומת המחלקה

שדה מקור בפועל (Virtual Source): ניתן להגדיר בסכימה שדה אשר ערכו יהיה זהה לערכו של שדה אחר. השדה אינו מופיע באופן ממשי בתוך בסיס הנתונים אלא מוצג רק לתוכנית היישום כאשר היא קוראת את הרשומה.

שגרות ספריה: שגרות הספריה הן שגרות מיוחדות הנכתבות על ידי האחראי על בסיס הנתונים ומופעלות באופן אוטומטי על ידי תוכנית הבקרה של מערכת DBMS בזמן העבודה. באמצעות שגרות אלה ניתן לבצע מטלות שונות, כגון:

— בדיקות מורכבות על שדות לפני הכנסתם לבסיס הנתונים,
— חישוב שדות בפועל,
— עיבוד נוסף לפני כתיבת רשומה חדשה, ביטול רשומה או עדכון,
— טיפול במצבי שגיאה מיוחדים,
— פונקציות מיוחדות (סטטיסטיקות למשל).

השגרות מנוהלות בספרית שגרות והן יכולות להיות כתובות בשפות שונות (שפת סף, קובול, פורטרן).

בדוגמא זו השגרה DELETE-PATIENT תופעל לפני שהמערכת תבטל את הרשומה PATIENT.

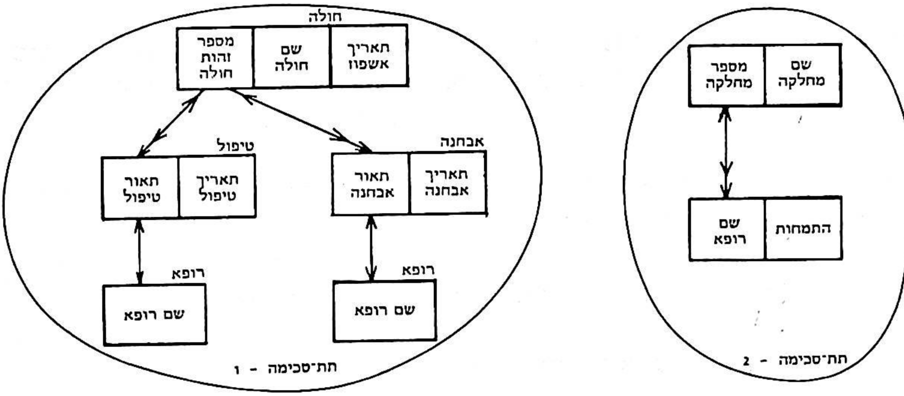
```
RECORD MAME IS PATIENT.
   ON INSERT CALL NEW-PATIENT-CHECK.
   ON  DELETE CALL DELETE-PATIENT.
   ON MODIFY CALL MODIFY-PATIENT.
```

4. תת־סכימה

תת־הסכימה (User View, External Schema, Subschema) הינה **התיאור הפורמלי** והמפורט של **מודל הנתונים של המשתמש**. היא מציגה את נקודת המבט של המשתמש או של תוכנית היישום ולכן מתארת רק קטע מתוך הסכימה אשר מעניין את המשתמש. המשתמש אינו צריך להכיר את כל הסכימה, אם בגלל מורכבותה הרבה ואם מסיבות של סודיות ובטיחות נתונים.

על בסיס סכימה אחת ניתן לבנות מספר רב של תת־סכימות, כל אחת מהן בהתאם לדרישות המידע היחודיות של תוכניות היישום השונות. תת־סכימה אחת יכולה לשרת מספר כלשהו של תוכניות יישום שונות.

הגדרת תת־סכימה אינה אפשרית בכל מערכות DBMS. יש מערכות שבהן תוכנית היישום מטפלת בבסיס הנתונים באמצעות הסכימה עצמה (מערכת TOTALלמשל).



תרשים 4 – 4: תת־סכימות

תרשים 4—4 מתאר שתי תת־סכימות שונות לחלוטין המבוססות על הסכימה שמוצגת בתרשים 2—4. תת־סכימה—1 מיועדת לשאילתות על האבחנות והטיפולים שקיבל חולה. תת־סכימה—2 מאפשרת דיווח של רשימת רופאים לפי מחלקות. כפי שניתן לראות כל תוכנית יישום רואה רק את החלק של בסיס הנתונים אשר דרוש לה.

בין הסכימה לתת־סכימה מותרים הבדלים מסויימים, לדוגמא:

—**ברמת השדה** —ניתן להתעלם משדות מסויימים,
—ניתן לשנות תכונות של שדה, למשל להפוך שדה עשרוני לבינרי,
—אפשרי סידור שונה של השדות ברשומה,
—אפשרי קיבוץ מספר שדות לשדה קבוצתי.

—**ברמת הרשומה** —ניתן להתעלם מרשומות מסויימות,
—ניתן לתאר רשומות חדשות המורכבות מחלק מהרשומות הקיימות.

—**ברמת הקשר** —ניתן להתעלם מחלק מהקשרים.

קיימות מערכות שאינן משתמשות בתת־הסכימה כמנגנון להגדרת נקודת המבט של תוכנית היישום. במערכות אלו תוכנית היישום מגדירה את השדות המבוקשים והפנייה לבסיס הנתונים היא **לפי שם השדה** (Field Request Method). חלק מהפרמטרים של פקודת הטיפול בנתונים הם שמות השדות, לדוגמא:

```
GET  DEPARTMENT RECORD USING
     DEP-ID
     DEP-NAME.
```

הוראה זו תגרום לכך שתוכנית הבקרה תעביר לתוכנית היישום רק את שני השדות המבוקשים.

5. סכימה פנימית

הסכימה הפנימית (Physical Schema, Internal Schema) מתארת את המבנה הפיסי של הנתונים כפי שהם מאוחסנים ביחידות האחסון השונות. הסכימה הפנימית מכילה פרטים הנוגעים לצורת האחסון של הקבצים, שיטות גישה (אינדקסים, גישה ישירה, מצביעים), מספר הרשומות שיש לאחסן בגוש בדיסק וכדומה.

הסכימה הפנימית משפיעה על טיב הביצועים של מערכת DBMS אך לא על הסכימה והתת־סכימות. ניתן ליישם סכימה אחת במספר סכימות פנימיות שונות ובכולן נקבל מבנה לוגי דומה. השוני ביניהן יהיה בטיב הביצועים של תוכניות היישום בשל השוני בארגון בסיסי הנתונים ושיטות הגישה אליהם.

קיימות מערכות DBMS שבהן אין הפרדה ברורה בין הסכימה לבין הסכימה הפנימית ולכן נמצא התייחסות למבנים פיסיים במסגרת הסכימה עצמה. הסיבה לכך היא שיש הטוענים שהפרדה בין שתי הסכימות האלה לא תאפשר אופטימיזציה של הארגון הפיסי של הנתונים, ולכן רמת הביצועים של המערכת עלולה להפגע. מאידך, יש הטוענים שהתייחסות למבנים פיסיים בסכימה פוגעת באי־תלות הנתונים. לדעתם המבנים הפיסיים תלויים רק בצורת השימוש בבסיס הנתונים, כמו בסוג ואופי השאילתות, ולא במבנה הלוגי של הנתונים.

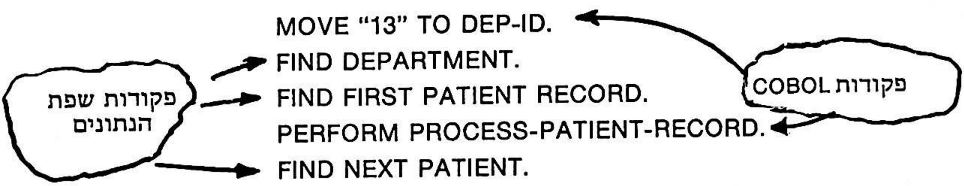
בהמלצות CODASYL מופיעה פשרה מסויימת בין שתי הטענות. ההמלצות מאפשרות התייחסות כלשהי לארגון הפיסי של הנתונים, אך לא התייחסות ישירה לתכונות פיסיות של יחידת האחסון (כגון גודל מסילה, גודל גוש וכדומה). בהמלצות מופיע המושג **שטח** או **איזור** (Area) שהינו חלוקה של בסיס הנתונים אשר מאפשרת הקצאת סוגי רשומות לשטחים מוגדרים. השטח מאפשר חלוקת בסיס הנתונים על פני יחידות האחסון לשיפור הביצועים. הגדרת **שטח** תהיה שונה ביחידות אחסון שונות ותהייה קרובה לתיאור התכונות הפיסיות של היחידה.

6. שפה לטיפול בנתונים

השפה לטיפול בנתונים (DML — Data Manipulation Language) הינה אוסף של פקודות המאפשרות לתוכנית היישום לאחסן, לשלוף ולעדכן את הנתונים שבבסיס הנתונים. השפה תלוייה בדרך כלל במודל הנתונים, ובעזרתה נותנת תוכנית היישום את ההוראות לתוכנית הבקרה על הפעולות שיש לבצע.

רוב מערכות DBMS הקיימות מכילות שני סוגי שפות נתונים. **שפת נתונים** המבוססת על שפה מארחת אשר מיועדת בעיקר לתוכניתני היישומים **ושפת שאילתות** המיועדות בעיקר למשתמשים המעוניינים בנתונים לצורך עבודתם (Business Professionals).

— **שפת נתונים המשמשת בשפת תכנות כלשהי כשפה מארחת** (Host Language). ניתן להתייחס אל שפת הנתונים כאל הרחבה של השפה המארחת, למשל COBOL או PL/1, והפקודות שלה מופיעות בין הפקודות הרגילות של השפה המארחת. פקודות אלה מתורגמות על ידי **קדם־מהדר** (Pre-Compiler) לפקודות CALL אשר מפעילות שגרות של תוכנית הבקרה לביצוע הפעולות הדרושות. לדוגמא,



— **שפת נתונים עצמאית המכילה הן פקודות לגישה לנתונים והן לעיבודם.** לעיתים מקובל לקרוא לשפות אלה **שפות שאילתות** (Query languages). הן מאפשרות למשתמש המנוסה או הפחות מנוסה לעבד את הנתונים באמצעות שפה פשוטה יחסית לשימוש אך בעלת עוצמה רבה. שפת השאילתות אינה מהווה ברוב המקרים חלק ממערכת DBMS ואינה כוללת פקודות לעדכון נתונים בבסיס הנתונים.

את הפקודות המרכיבות את שפת הנתונים אפשר לסווג בשלש קבוצות:

— **פקודות בקרה:** באמצעות פקודות אלה מזדהה תוכנית היישום ומגדירה את בסיס הנתונים שבו הוא רוצה להשתמש. קבוצה זו כוללת פקודות כגון OPEN, CLOSE ואפשרויות להגדיר את סוג העיבוד של התוכנית (קריאה בלבד, קריאה ועדכון וכד')

— **פקודות לאחזור נתונים:** באמצעות פקודות אלה מגדיר המשתמש את הרשומות המבוקשות. לעיתים מתבצע האחזור על ידי גישה לרשומה מסויימת ולעיתים האחזור מחייב סריקת רשומות וקשרים באופן נסתר מהמשתמש. באופן עקרוני קיימות שתי צורות אחזור בסיסיות; **אחזור לפי תוכן** (Contents Addressability) **ואחזור לפי קשר בין הרשומות** (Data Relatability). האחזור לפי תוכן מאפשר למשתמש לאחזר רשומה בהתאם לתוכן של אחד מהשדות שלה כמו למשל אחזור רשומות המחלקה לפי מספר המחלקה.

שם רשומה

FIND DEPARTMEMT RECORD
WHERE DEP—ID=13.

תנאי (Qualification)

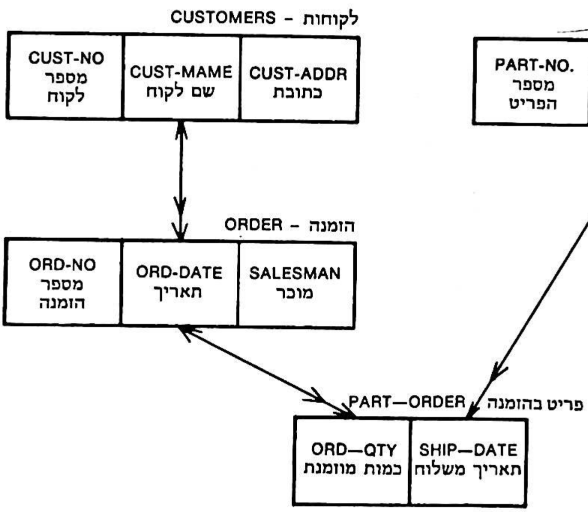
בדרך כלל האחזור לפי תוכן מלווה בתנאי כלשהו הקובע את זהות הרשומה המבוקשת. קיימת אפשרות לבניית תנאי מורכב יותר כמו למשל לאחזור רשומת מחלקה שבה מועסקים יותר מ־20 רופאים ויש בה לפחות 30 מיטות תפוסות:

```
FIND DEPARTMENT RECORD
  WHERE NUMB—PHYSICIANS > 20
AND WHERE OCCUPIED—BEDS NOT < 30.
```

האחזור לפי קשר בין רשומות מאפשר אחזור רשומה אחת או יותר הקשורות לרשומה מסויימת. לדוגמא, לאחר שזיהינו את רשומת המחלקה שמקיימת את התנאי הנזכר נרצה לאחזר את רשומת החולה הראשון המאושפז במחלקה. נעשה זאת כך:

FIND FIRST PATIFNT RECORD.

האחזור לפי קשר בין רשומות תלוי במודל הנתונים. נשתמש בדוגמא המבוססת על מודל הנתונים הרשתי שבתרשים 5—4.



תרשים 5 - 4: מודל נתונים לאחזור רשומות

אפשר לאחזר רשומות מתוך בסיס הנתונים באופנים אחדים:

א. אחזור רשומת פריט שמספרו A129. זוהי דוגמא לאחזור לפי תוכן בלבד.

```
FIND PART  RECORD
     WHERE  PART—NO=A129.
```

ב. אחזור כל ההזמנות של לקוח ששמו ABC.

```
FIND CUSTOMER RECORD            <---- אחזור לפי תוכן
     WHERE CUST—NAME=ABC.
FIND FIRST ORDER RECORD.        <---- אחזור לפי קשר
PERFORM PROCESS—ORDER.                בין רשומות
FIND NEXT ORDER RECORD.         <----
```

— **פקודות לעדכון נתונים:** באמצעות פקודות אלה מעדכן המשתמש את בסיס הנתונים. לרשותו פקודות המאפשרות הוספת רשומות חדשות, ביטול רשומות קיימות ועדכון תוכן רשימות קיימות. לדוגמא, ביטול רשומת מחלקה:

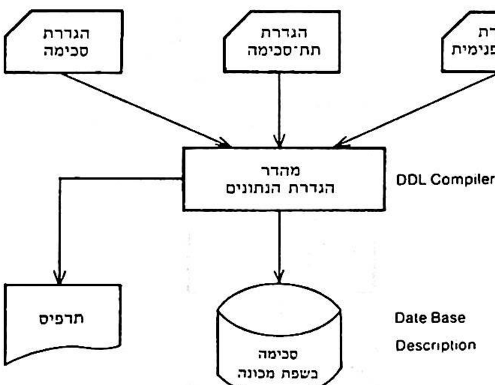
```
DELETE DEPARTMENT RECORD.
```

7. עקרונות הפעולה של מערכת DBMS

בסעיף זה נציג את שיטת העבודה של מערכת DBMS החל משלב הגדרת הנתונים ועד גישה לנתונים מתוך תוכנית יישום.

7.1 מהדר השפה להגדרת נתונים (Data Definition Language Compiler)

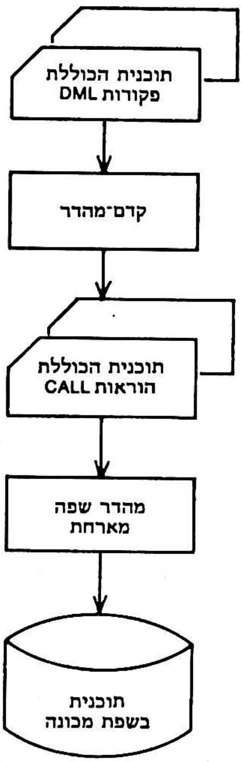
רוב מערכות DBMS המשתמשות בסכימות השונות כפי שהוצגו בסעיף קודם מספקות מהדר מיוחד שמיועד לבניית הסכימות. מהדר זה מקבל כקלט את הפקודות המיוחדות של השפה להגדרת הנתונים והופך אותן לפקודות בשפת מכונה ולאוסף של טבלאות שתשמשנה לאחר מכן את תוכנית הבקרה לביצוע המיפויים השונים.
במספר מערכות יש מהדר נפרד לכל אחת מהסכימות.



תרשים 6 – 4: מהדר הגדרת נתונים

7.2 קדם־מהדר של שפה מארחת (Host-Language Pre-compiler)

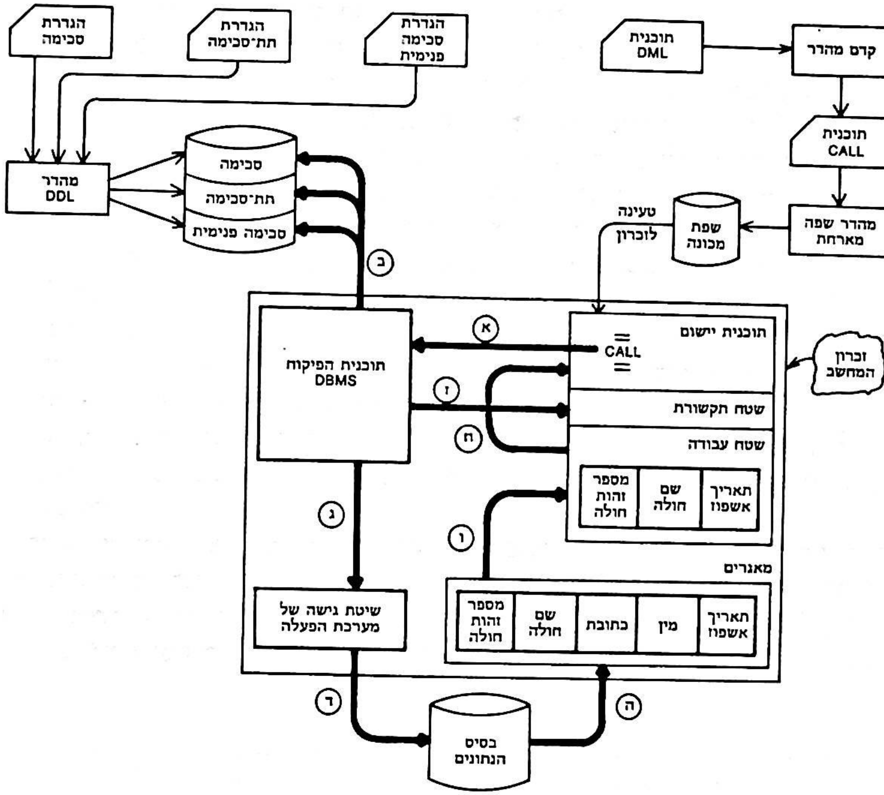
תוכנית היישום משתמשת **בפקודות טיפול בנתונים** (DML-Data Manipulation Language) אשר שייכות למערכת DBMS ואינן מוכרות על־ידי המהדר של השפה המארחת. מערכות DBMS מספקות קדם־מהדר אשר קורא את תוכנית היישום שבה ישנן פקודות טיפול בנתונים (DML) ומתרגם אותן להוראות CALL אשר מוכרות על־ידי המהדר של השפה המארחת. קדם־המהדר קובע את הפרמטרים השונים של הוראות CALL בהתאם לפקודות DML המקוריות. הוראות CALL מפעילות את תוכנית הבקרה של מערכת DBMS בעת שתוכנית היישום רוצה לגשת לנתונים.



תרשים 7 – 4: השלבים בתהליך ההידור של תוכנית היישום

7.3 עקרון הפעולה

המודל הכללי של מערכת DBMS המבוססת על שפה מארחת יוצג באמצעות תיאורי הסכימה ותת־סכימה שמופיעים בתרשימים 2—4, 4—4 ובתרשים המודל 8—4.



תרשים 8 – 4: מודל להצגת עקרון פעולת מערכת DBMS

בתרשים 8—4 מוצג בצורה גרפית עקרון פעולת מערכת DBMS. בתחילה יש להגדיר את הסכימות ולהפעיל את המהדרים של שפת הגדרת הסכימה (DDL) ושל שפת הטיפול בנתונים (DML), ואת המהדר של שפת תוכנית היישום. לאחר טעינת התוכנית לזכרון היא מתחילה בפעולתה, אשר תוצג בשלבים (לפי הסימולים בתרשים):

א. תוכנית היישום מבקשת ממערכת DBMS להציג רשומת חולה אשר מספרו 145. התוכנית מבקשת רשומה לוגית אשר המבנה שלה מוגדר בתת־סכימה המתאימה לתוכנית היישום. מערכת DBMS מופעלת באמצעות הוראת CALL אשר בגוף תוכנית היישום.

ב. תוכנית הפיקוח של מערכת DBMS מנתחת את הבקשה ומשתמשת לשם כך בסכימה ובתת־סכימה המתאימות. המיפוי בין התת־סכימה לסכימה מורה לתוכנית הפיקוח שהתוכנית מבקשת את רשומת החולה ומציג לתוכנית הפיקוח את המבנה הפיסי של קובץ "חולים".

ג. תוכנית הפיקוח "מנסחת בקשה" לשגרות שיטת־גישה (Access Method) של מערכת ההפעלה. הבקשה כוללת את כל הפרמטרים הדרושים למערכת ההפעלה כדי לבצע את

הגישה ליחידת האחסון. רוב הפרמטרים מתקבלים מהסכימה הפנימית.

ד. שגרות מערכת ההפעלה מבצעות את הגישה לנתונים המאוחסנים ביחידת האחסון החיצונית. השגרות מטפלות בגושים ולא ברשומות.

ה גוש שמכיל את הרשומה המבוקשת מועבר אל שטח מאגר קלט/פלט (I/O Buffer). השגרות מחזירות את הפיקוח לתוכנית הפיקוח של מערכת DBMS אשר מוצאת במאגר גוש שמכיל את הרשומה המלאה כפי שהיא מופיעה בסכימה.

ו. תוכנית הפיקוח משתמשת שוב בסכימה ובתת־סכימה כדי לבצע את המיפוי הדרוש בין הרשומה כפי שהיא מופיעה במאגר לבין הרשומה המבוקשת על־ידי תוכנית היישום. תוכנית הפיקוח מאתרת בגוש את הרשומה המבוקשת ומעבירה לשטח העבודה בתוכנית היישום רק את השדות המבוקשים. בתרשים ניתן לראות שבשטח העבודה מוצגת רשומה לוגית שאינה קיימת באותו מבנה בבסיס הנתונים.

ז. תוכנית הפיקוח מעבירה לשטח התקשורת שבינה לבין תוכנית היישום מספר נתונים אשר כוללים בין השאר **מציין־מצב** (Status Indicator). מציין־מצב מורה את אופן סיום הוראת קריאת הנתונים: הרשומה בשטח העבודה, רשומה לא קיימת וכדומה.

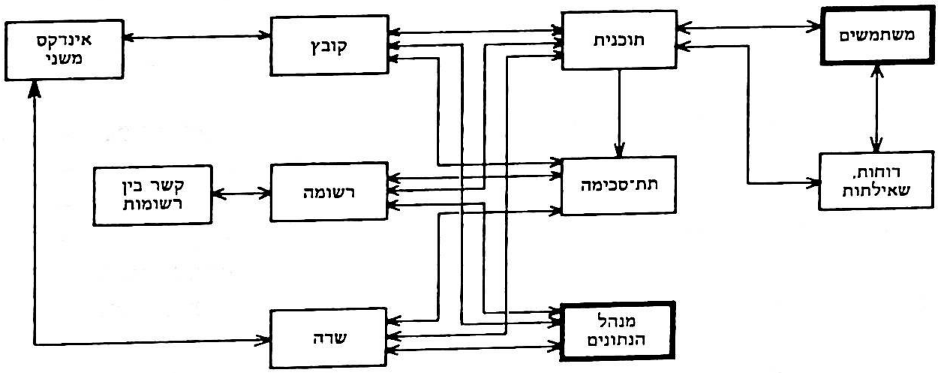
ח. תוכנית הפיקוח של מערכת DBMS מחזירה את הפיקוח לתוכנית היישום. לאחר בדיקת מציין המצב שבשטח התקשורת מתחילה תוכנית היישום את עיבוד הרשומה שבשטח העבודה.

כפי שניתן לראות מתיאור זה, אי־התלות בנתונים נוצר באמצעות שני המיפויים: המיפוי שבין תת־הסכימה לבין הסכימה והמיפוי שבין הסכימה לסכימה הפנימית. לדוגמא, הוספת שדה חדש ברשומת "חולה" אינה משפיעה על תוכנית היישום ששדה זה אינו דרוש לה. השדה אינו מוגדר בתת־הסכימה השייכת לתוכנית זו ועל כן לא יועבר לשטח העבודה שלה.

8. מילון נתונים

עלות עיבוד הנתונים הינה פונקציה ישירה של טיב ניהול הנתונים מכיון שהנתונים הינם משאב ככל משאב אחר בארגון. ככל שכמות הנתונים במערכת המידע גדולה יותר יש יותר גורמים המטפלים בהם ולכן נוצר סיכון רב יותר משימוש לא מבוקר בבסיס הנתונים. ניהול לקוי של הנתונים יכול לגרום לכפילות נתונים רבה, מאמצי תכנות מיותרים ועוד, ובסופו של דבר — לאבדן זמן וכסף בפיתוח ובאחזקה של מערכות המידע. כדי לסייע בתחום זה פותח רעיון **מילון הנתונים** (Data Dictionary) אשר מיועד למנהל בסיס הנתונים ולמשתמש בבסיס הנתונים.

מילון הנתונים מכיל את **המידע על המידע**, כלומר נתונים המתייחסים ליחידות הלוגיות המאוחסנות בבסיס הנתונים כמו שמות ותכונות. בעזרת מילון הנתונים ניתן לדעת מהם הנתונים המאוחסנים בבסיס הנתונים ומהן תכונותיהם, אילו תוכניות משתמשות באילו נתונים, ועוד. מילון הנתונים יכול להקטין את ההשקעה הדרושה באחזקה ובפיתוח יישומים, בעיקר בגלל האחידות והריכוזיות של המידע על המידע. קיימות מערכות DBMS המשתמשות במילון הנתונים כסכימה ותת־סכימה. קיימות מערכות שמשתמשות בבסיסי נתונים שלהן לניהול המילון, ויש מערכות שבהן המילון הוא חלק נפרד ואף מנוהל במודל נתונים שונה.



תרשים 9 – 4: יחידות לוגיות וקשרים במילון נתונים

תרשים 9—4 מציג את היחידות הלוגיות השונות שניתן לנהל במילון נתונים. לדוגמא, ניתן לנהל מידע על כל התוכניות שמשתמשות בשדה נתונים כלשהו או על המשתמשים בתוכנית יישום כלשהי, אילו דוחות מקבל כל משתמש, מי אחראי לעדכון הקובץ או סוגי רשומה, ועוד.

8.1 הנתונים במילון

מילון הנתונים יכול להכיל נתונים רבים ומגוונים בהתאם לשימוש אשר נעשה בו. לדוגמא, הוא יכול להכיל את הנתונים הבאים:

רמת השדה:

— שם השדה,
— אורך השדה,
— סוג השדה (נומרי, אלפאביתי, מספר שלם, מספר בנקודה צפה),
— תחום ערכים,
— תיאור מילולי של תוכן השדה,
— מבנה השדה לצורך עריכה בדוחות,
— כותרת שדה (סטנדרטית) בדוחות שונים,
— מבנה השדה בקלט,
— נתוני מצב (מתי עודכן לאחרונה, בשימוש או לא, ועוד),
— אילו תוכניות קוראות אותו או מעדכנות אותו.

רמת הרשומה:

— שם הרשומה,
— אורך הרשומה,
— תיאור מילולי של תוכן הרשומה,
— אילו תוכניות כותבות רשומה חדשה, מבטלות או מעדכנות.

רמת הקובץ:

— שם הקובץ,
— מספר רשומות (מספר מדוייק או מקורב),
— תיאור מילולי,
— מפתח הקובץ,
— שיטת הארגון,
— סדר המיון,
— גודל הגוש,
— אילו תוכניות משתמשות בקובץ.

רמת התוכנית:

— שם התוכנית,
— שם המחבר,
— תאריך כניסה לייצור,
— שם תת־סכימה שהתוכנית משתמשת בה,
— תיאור מילולי של מבנה ותפקיד התוכנית,
— סימול המשתמשים בתוכנית.

רמת תת־סכימה:

— שם תת־הסכימה,
— אילו תוכניות משתמשות בה,
— אילו קבצים, רשומות ושדות מוגדרים בתת־סכימה,
— תיאור מילולי.

8.2 השירותים של מילון הנתונים

בעזרת מילון הנתונים ניתן לדעת באילו תוכניות יש לבצע הידור מחדש לאחר שינוי בתת־סכימה או סכימה. מתוך מידע זה יכול המשתמש להעריך את העלות הצפוייה לביצוע השינוי.

רוב המערכות המפעילות מילון נתונים מספקות שפת שאילתות אשר מאפשרת שליפת נתונים קלה מתוך מילון הנתונים והפקת דוחות מהירה ופשוטה. במערכת SQL/DS למשל משתמשים למטרה זו בשפת השאילתות של המערכת אשר מיועדת לשאילתות בבסיס הנתונים.

השימוש במילון הנתונים נעשה על ידי גורמים שונים ביחידות עיבוד הנתונים וכל אחד מהם יכול להפיק מכך יתרונות שונים:

משתמש במערכת המידע:

— הבנה טובה של הנתונים המאוחסנים בבסיס הנתונים ומשמעותם (לרוב ניתן גם תיאור מילולי),
— כלי תיעודי גם לנתונים לא טכניים הקשורים במערכת המידע,
— בסיס לתקשורת עם יחידת עיבוד הנתונים,
— מידע באילו נתונים הוא משתמש ואילו נתונים הוא מייצר,
— סיווג הנתונים לפי ה״בעלים״ שלהם בארגון (פרטים או מחלקות).

מנתח המערכות:

— בדיקת אלטרנטיבות ליישום מערכת חדשה,
— בדיקת עלות השינוי הצפויה לפני ביצוע שינוי כלשהו במערכת,
— הגדרה נוחה של השימוש בנתונים ושימוש חוזר בהגדרות לאחר מכן עבור תוכניות אחרות,
— כלי תיעודי לנתונים טכניים אודות בסיס הנתונים והתוכניות.

מתכנת:

— חסכון בזמן פיתוח תוכנה מכיון שניתן להשתמש בהגדרות קיימות של נתונים ותוכניות,
— סיוע למתכנתים חדשים להכיר את מבנה הנתונים והקשרים ביניהם,
— כלי תיעוד אוטומטי לתוכניות ונתונים.

מנהל בסיס הנתונים:

— בקרה ושליטה על הנתונים שבבסיס הנתונים,
— הקטנת כפילות הנתונים,
— אכיפת סטנדרטים לנתונים ולתוכנה,
— "חקירה" מהירה של רכיבי בסיס הנתונים.

9. שפת שאילתות

רוב מערכות DBMS משתמשות בשפת שאילתות (Query Language) לא פרוצדורלית אשר המבנה שלה הוא דמוי שפה אנגלית טבעית. מטרת שפת השאילתות היא לאפשר למשתמשים גישה נוחה לנתונים המאוחסנים בבסיס הנתונים. היא מאפשרת היענות מהירה לדרישות מזדמנות לנתונים ומונעת את הצורך לכתוב תוכניות מחשב בנוהל המקובל.

שפת השאילתות מאפשרת גישה לנתונים על פי תנאי שליפה לוגיים מורכבים וניסוחה הינו למעשה תכנות של דרישת המשתמש לנתונים. קיימות שפות שאילתות שונות אשר נבדלות זו מזו באפשרות ניסוח הדרישה או השאילתא, אפשרות עריכת הפלט, יעילות אחזור הנתונים, אפשרות עדכון נתונים ועוד.

המשתמש בשפת שאילתות להפקת נתונים מבסיס הנתונים חייב לזכור שהשאילתא הינה תוכנית אשר מורכבת משלבים של סריקת אינדקסים ונתונים, השוואות ובדיקות לוגיות שונות ולבסוף ריכוז הנתונים ועריכת הדוח. זמן התגובה לביצוע שאילתא שונה מזמן תגובה לאחזור רשומה לקובץ מכיון שמסלול הפעילות שונה בשני המקרים. ישנן שפות שאילתות הקשורות בבסיס נתונים אשר מאפשרות רמת יעילות גבוהה בביצוע האחזור והפקת הדוח המבוקש גם בקבצים בנפחים של מאות־אלפי רשומות. עם זאת, השאילתות מיועדות לדרישות אקראיות מזדמנות ולא לתהליכי העיבוד העיקריים של יישומי המערכת.

10.שאלות

1. לרוב מערכות DBMS המסחריות יש שפה מארחת. מדוע?
2. נסה לתכנן שפת שאילתות ה"תפורה" במיוחד עבור שאילתות במערכת הזמנת כרטיסי טיסה.
3. אחד ההבדלים בין תת־סכימה לסכימה יכול להיות בסוג השדה. הסבר האם דבר זה רצוי ומה הקשיים בביצוע הסבה אוטומטית של סוג השדה.
4. הסתכל על תוכנית COBOL רגילה. נסה לזהות את מרכיבי DDL, DML.
5. הסכימה ותת־הסכימה עבור בסיס נתונים מסויים מבוססים על אותו מודל נתונים. מהם החשיבות והקשיים להציג תת־סכימה המבוססת על מודל נתונים שונה מהסכימה.
6. מדוע רצוי שסכימה תהיה משוחררת מהתייחסות לארגון הפיסי של הנתונים. האם היא יכולה להיות משוחררת לחלוטין?
7. כאשר מערכת DBMS מופעלת בפיקוח מערכת הפעלה רגילה, האם מערכת ההפעלה "יודעת" מה נמצא בתוך הגושים שהיא מאחסנת ושולפת מיחידות אחסנה חיצוניות? האם היא יודעת אם הגוש מכיל נתונים או הוראות בקרה?

פרק ה

מודל נתונים היררכי
(Hierarchical Data Model)

1. מבוא
2. המודל ההיררכי – הגדרה
3. ייצוג קשרים N:M במודל נתונים היררכי
4. שיטות לייצוג מבנה היררכי
5. שפה לטיפול במבנה היררכי
6. שאלות

1. מבוא

בהתפתחות מודלים של נתונים יש למודל ההירככי תפקיד חשוב מאד. בסיס נתונים הירככי בנוי לפי מודל פשוט יחסית והוא עונה על צרכיהם של רוב המשתמשים. מודל המבוסס על הירככיות ניתן ליישום ישיר באמצעות טכניקות ארגון קבצים המבוססות על מבני עץ. היתרון של בסיס נתונים זה הוא השימוש במסלולי גישה מוגדרים מראש המיושמים בבסיס הנתונים. אין צורך בבחירת מסלולים מורכבת בזמן עיבוד הקובץ (קריאה, כתיבה, הוספה, ביטול) ובעת ביצוע שאילתות, ולפיכך רמת הביצוע טובה.

בפרק זה נציג את מודל הנתונים ההירככי בצורה מפורטת ונתייחס למבנה השפה ההירככית המאפשרת את הטיפול במבנים הירככיים. חלק ניכר מתשומת הלב לה זכה מודל זה נובעת מהעובדה שמערכות DBMS הראשיות של חברת יבמ (DL/1) מבוססות על מודל זה. את מערכת IMS נציג במפורט בחלק השני של הספר. קיימות מערכות מסחריות נוספות המתבססות על המודל ההירככי כמו למשל SYSTEM 2000 של חברת Intel, MARK IV של חברת Informatics.

2. המודל ההירכי – הגדרה

המודל ההירכי מבוסס על אוסף של סוגי רשומות ושל קשרים בין סוגי רשומות אלו. הרשומות מייצגות את הישויות ואת התכונות שלהן, והקשרים בין סוגי הרשומות מייצגים את הקשרים שבין הישויות. להלן הגדרה פורמלית של המודל ההירכי:

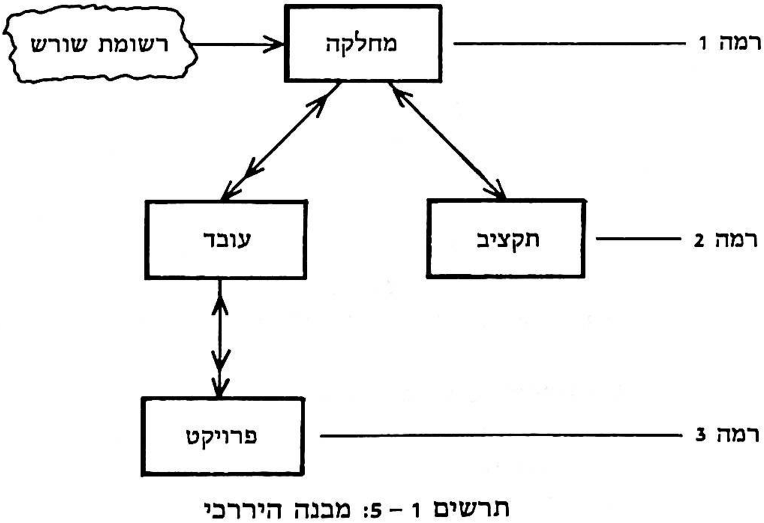
א. קיים אוסף של סוגי רשומות [R1, R2, R3, ...Rn]. לכל סוג רשומה מספר בלתי מוגבל של מופעים בבסיס הנתונים.

ב. קיים אוסף של קשרים, Lij, הקושרים את סוגי הרשומות למבנה נתונים הירכי אחד. הקשרים מקיימים את הכללים הבאים:

— קשר Lij אפשרי בין שני סוג רשומות Rj, Ri רק אם Ri נמצא ברמה הירכית n ו־Rj ברמה הירכית n+1.
— בין כל שני סוגי רשומות Ri ו־Rj קיים קשר Lij אחד לכל היותר. מכאן נובע שאין צורך לתת שמות לקשרים בין סוגי הרשומות.
— הקשר Lij הינו פונקציונלי בכיוון אחד. לכל מופע של רשומה מסוג j יש בדיוק מופע אחד של רשומה מסוג i הקשורה אליה. בכיוון השני הקשר אינו פונקציונלי. כלומר, לכל מופע של רשומה מסוג i יש מספר מופעים של רשומות מסוג j הקשורות אליה. לסוג רשומה j נקרא **רשומת אב** ולסוג רשומה j נקרא **רשומת בן**.
— לא ניתן להגדיר קשר מסוג Lii. כלומר, אין אפשרות להגדיר קשר בין מופעים שונים של אותו סוג רשומה.
— לא ניתן להגדיר קשרים Lij ו־Lkj כי לכל סוג רשומה יש סוג רשומת אב אחת בלבד.

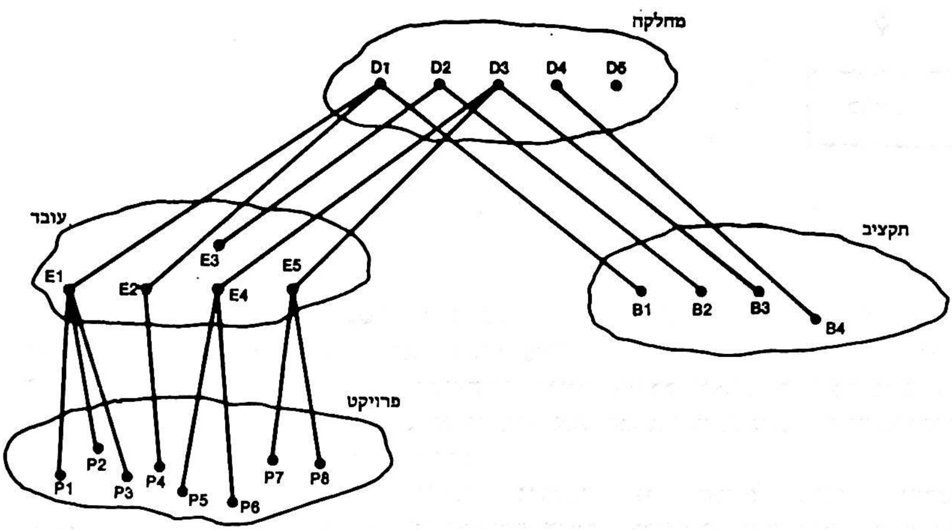
ג. בין כל סוגי הרשומות קיים סוג רשומה מיוחד — **רשומת שורש** (Root Record) שאינה קשורה לשום רשומת אב, אך יכולות להיות לה רשומות בן.

נשתמש בדוגמאות אחדות כדי להסביר את המודל ההירכי. מתוך ההגדרה נובע שהמודל מורכב מסוגי רשומות וקשרים מסוג 1:1 או 1:N בלבד.



תרשים 1 – 5: מבנה הירכי

תרשים 1—5 מציג מבנה עץ בו כל **צומת** מייצג סוג רשומה שונה ואילו **הקשתות** המחברות את סוגי הרשומות מייצגות את הקשרים בין סוגי רשומות. בין כל שני צמתים יש קשר אחד לכל היותר ולכן אין צורך לתת שמות לקשרים. רשומת השורש אינה תלויה ברשומות אחרות והיא היחידה שיש לה קיום עצמאי. כל שאר סוגי הרשומות הן תלויות ויש להם זכות קיום רק בהיותן קשורות לרשומת השורש. בדוגמא שבתרשים 1—5 רשומות המחלקה יכולות להתקיים גם אם אין רשומות אחרות הקשורות אליהן, אך רשומת עובד יכולה להתקיים רק אם היא קשורה לרשומת מחלקה כלשהי.



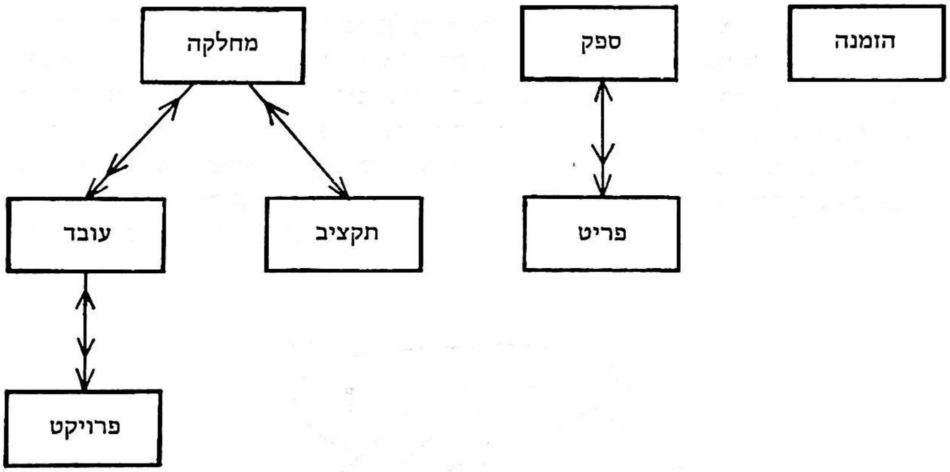
תרשים 2 - 5: מופעים של רשומות במבנה הירככי

באופן כללי, בסיס נתונים הירככי הינו אוסף של מספר בלתי מוגבל של מופעים של רשומות הקשורות ביניהן במבנה הירככי. בתרשים 2—5 ניתן לראות מספר מופעים של רשומות מחלקה למשל אשר קשורות למופעים של רשומות עובד ורשומות תקציב. במחלקה D4 אין עדיין עובדים אך כבר נקבע עבורה תקציב. רשומת מחלקה D1 קשורה לרשומות עובד E1, E2

עובד E3, אינו מועסק כעת בפרויקט ולכן אינו קשור לרשומת פרויקט כלשהי. כפי שניתן לראות, אין מופעים עצמאיים של רשומות במבנה ההירככי למעט המופע של רשומה D5 שהינה רשומת שורש. לפי כללי המודל אסור קשר בין רשומות תקציב לרשומות פרויקט למשל, כי במקרה זה תהיינה לרשומת פרויקט שתי רשומות אב שונות, דבר שנוגד את הגדרת המודל.

ניתן להשתמש במספר מבנים הירככיים שונים במסגרת אותו בסיס נתונים כדי לייצג קשרים שונים בין סוגי רשומות זרות. בתרשים 3—5 מוצג אוסף של עצים הירככיים שונים הכלולים בבסיס נתונים אחד — זהו **יער** (Forest).

העצים השונים ביער יכולים להיות שונים במבנה ובמספר רמות ההירככיה שלהם.



תרשים 3 – 5: אוסף של מבנים הירככיים – יער

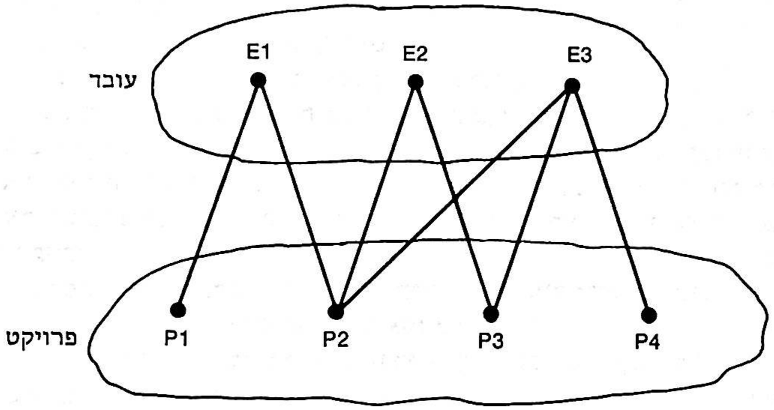
במודל ההירככי ניתן לזהות קשרי אב־בן בין כל שני סוגי רשומות הקשורות ביניהן פרט לרשומת השורש שאין לה רשומת אב. לדוגמא, בקשר בין מחלקה לעובד, רשומת המחלקה היא רשומת האב ורשומת העובד היא רשומת בן. המודל ההירככי מאפשר לרשומת בן בקשר אחד להיות רשומת אב בקשר אחר. רשומת העובד למשל, המוצגת כרשומת בן בקשר מחלקה־עובד הינה רשומת אב בקשר עובד־פרויקט.

הגישה לרשומות במודל ההירככי מבוססת על **מסלול גישה הירככי** (Hierarchical Access Path) המוביל מרשומת השורש לרשומת היעד באמצעות אוסף מסודר של רשומות הנמצאות בקשרי אב־בן ביניהן. המסלול ההירככי אל רשומת פרויקט יהיה מחלקה־עובד עובד־פרויקט.

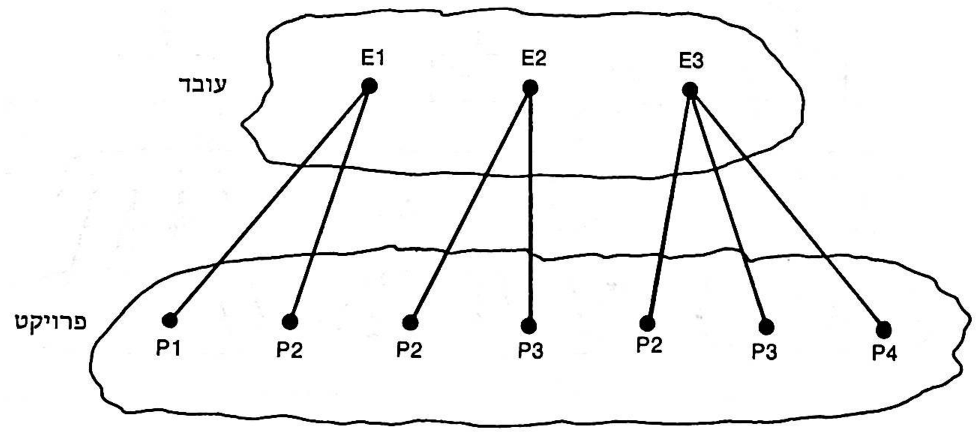
במקרים רבים הצגת הנתונים במבנה ההירככי הינה טבעית ופשוטה ולכן גם מובנת. היא מאפשרת תקשורת נוחה בין המשתמש או בעלי הנתונים, לבין יחידת עיבוד הנתונים.

3. ייצוג קשרים N:M במודל נתונים היררכי

המודל ההיררכי אינו מאפשר קשרים מסוג N:M והדבר גורם לקשיים מסויימים בהפעלת המודל. לדוגמא, אם עובד אחד מועסק במספר פרויקטים שונים ובפרויקט מועסקים מספר עובדים נאמר שקיים קשר רב—רב־ערכי בין העובד לפרויקט.



תרשים 4 – 5: קשר רב – רב־ערכי בין עובד לפרויקט



תרשים 5 – 5: ייצוג ישיר של קשר רב – רב־ערכי במבנה היררכי.

3.1 ייצוג ישיר של קשר N:M

הייצוג הישיר של קשר כזה במודל ההירככי מחייב כפילות נתונים מסויימת: שמירת הנתונים על הפרויקט עבור כל רשומת עובד המשתתף בו. בתרשים 5—5 ניתן לראות שרשומת הפרויקט P2 למשל, מופיעה שלש פעמים בבסיס הנתונים. כפילות נתונים זו מקשה על ניהול בסיס הנתונים מאחר ושינוי כלשהו בנתוני הפרויקט חייב להתבטא במספר רב של רשומות זהות. כדי להתגבר על כך פותחו דרכים לייצוג שונה של קשר רב—רב־ערכי במודל ההירככי.

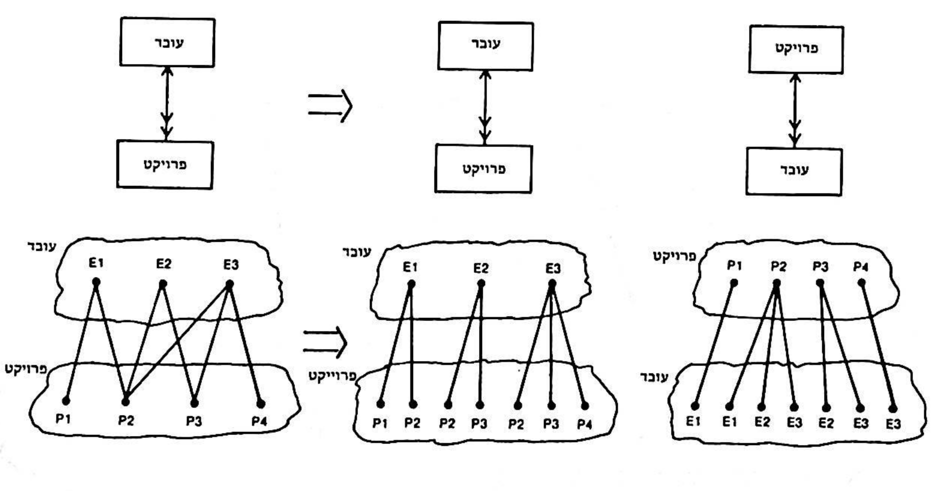
נעיין בשתי שאילתות שנראות סימטריות מבחינת המשתמש:

א. רשימת כל העובדים בפרויקט מסויים.

ב. רשימת כל הפרויקטים שבהם מועסק עובד מסויים.

הטיפול בשתי שאילתות סימטריות אלה **אינו סימטרי** במבנה שהוצג בתרשים 5—5. על שאילתא ב ניתן לענות בצורה ישירה כי מבנה הנתונים מציג את הקשר בצורה טבעית. על שאילתא א לא ניתן לענות בצורה ישירה. לשם כך יש לסרוק את כל רשומות הפרויקט, לאתר מתוכן את רשומות הפרויקט המבוקש ולזהות את רשומות האב שלהן. אי הסימטריה נובע מהעובדה שרשומות האב הן רשומות העובד ורשומות הבן הן רשומות הפרויקט. הדבר גורם לקושי מיותר מבחינת המשתמש, אשר צריך להקדיש זמן ומאמץ מיותרים לטיפול בבעיות אשר נגרמות על־ידי מודל שאינו מייצג את המציאות בצורה נוחה.

במקום ייצוג ישיר של קשר רב—רב־ערכי ניתן לייצג את הקשר על־ידי שני מבנים הירככיים זרים שכל אחד מהם מייצג קשר חד—רב־ערכי אחד. בתרשים 6—5 מוצג הקשר הרב—רב־ערכי בין העובד והפרויקט כשני קשרים חד—חד־ערכיים. בקשר אחד רשומת השורש היא רשומת העובד ובשני — רשומת השורש היא רשומת הפרויקט.

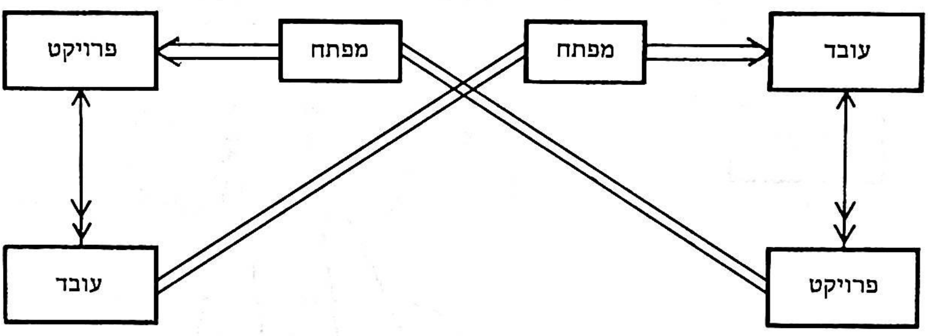


תרשים 6 - 5: הפיכת מבנה רשת ליער של עצים הירככיים

3.2 קשר אסוציאטיבי

הפיכת מבנה הרשת למבנה הירכי בתרשים 5—6 גרמה **לכפילות נתונים** רבה בבסיס הנתונים. רשומת פרויקט P2 מופיעה ארבע פעמים כרשומת בן ופעם נוספת כרשומת שורש במבנה הירכי אחר. אם כמות הנתונים שיש לאחסן לכל פרויקט היא רבה, יש בזבוז רב במקום האחסון. אך הבעייה העיקרית היא הצורך לעדכן רשומה אחת פעמים אחדות בגלל מספר המופעים שלה.

פתרון אפשרי לבעייה זו הוא בשמירת כל נתוני הפרויקט רק ברשומת השורש המתאימה. ברשומות הבן יש לשמור רק את מפתח הפרויקט ואת הנתונים הרלוונטים לקשר בין העובד לפרויקט כגון תאריך תחילת העבודה, שעות עבודה שהעובד השקיע בפרויקט וכדומה. בכך יוצרים **הצלבה של הנתונים** (Intersection Data) או **קשר אסוציאטיבי** (Associative Link) בין המבנים כפי שמוצג בתרשים 7—5.



תרשים 7 - 5: קשר אסוציאטיבי בין שני מבנים הירכיים

הקשר האסוציאטיבי בין שני המבנים ההירכיים מתאפשר בגלל אחסון המפתחות המתאימים בכל אחד מהם. הגישה מרשומת־בן "פרויקט" לרשומת־אב "פרויקט" מחייבת הפעלת שיטה כלשהי לתרגום מפתח לכתובת (על־ידי חישוב או באמצעות אינדקס) ולכן גישה זו איטית יותר מגישה בין רשומות השייכות לאותו מבנה הירכי. עדיין קיימת בעייה של חוסר סימטריה. קבלת שם הפרויקט למשל הינה מיידית אם נמצאים בעץ פרויקט־עובד כי הנתון הזה נמצא ברשומת השורש. לעומת זאת, אם נמצאים בעץ עובד־פרויקט יש לבצע גישה מיוחדת לרשומת השורש במבנה השני כדי לקבל את הנתון. מצב זה גורם לכך שקיימת נטייה להעביר חלק מהנתונים המופיעים ברשומת השורש גם לרשומות הבן במבנה ההירכי השני כדי לשפר את הביצועים על חשבון כפילות נתונים.

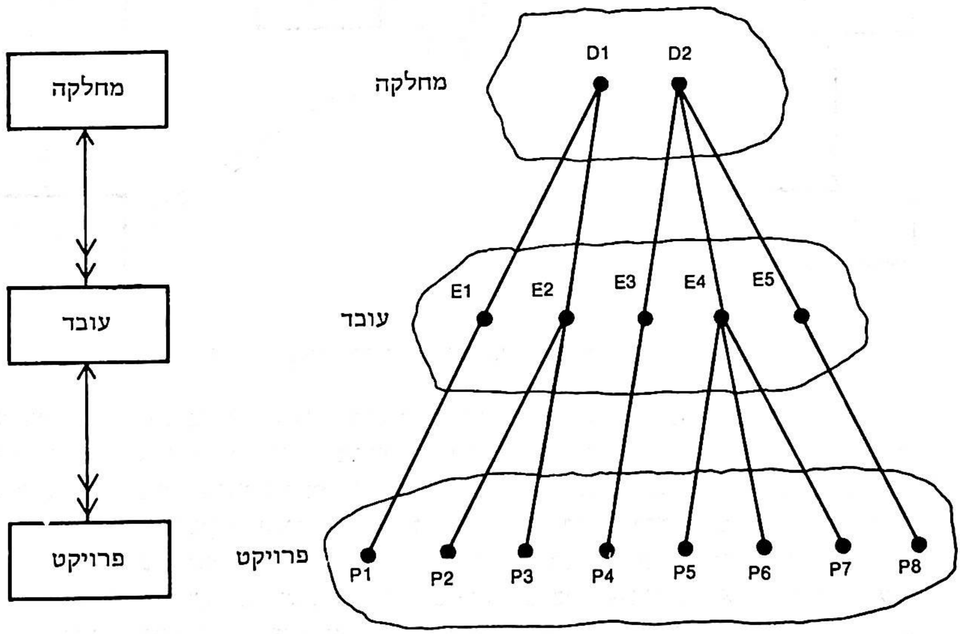
קיימות מערכות DBMS המאפשרות **קישור מבנים הירכיים** ביניהם (Interlinked Hierarchies) בצורה אוטומטית כמו למשל במערכת IMS של יבמ. הדבר נעשה על ידי שמירה נפרדת של רשומות בסיס הנתונים ושל טבלת הקשרים ביניהן.

טבלת הקשרים מכילה את הכתובות של הרשומות הרלוונטיות בלבד ולא את התוכן שלהם. הרשומות נמצאות על כן **פעם אחת בלבד** בבסיס הנתונים והבעיות של כפילות נתונים ועדכון לא קיימות כאן. תיאור מפורט של המערכת בחלק השני של הספר.

4. שיטות לייצוג מבנה היררכי

את מבנה העץ במודל הנתונים ההיררכי אפשר להציג בשתי דרכים.

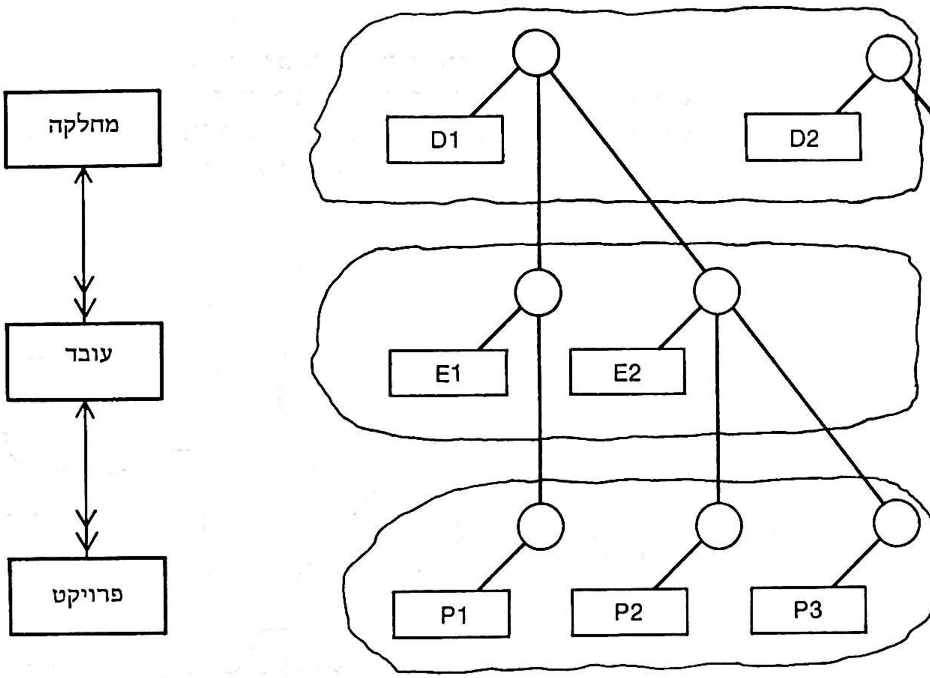
בשיטה הראשונה מוצגים הצמתים בעץ כמופעים של רשומות. בשיטה זו כל צומת במבנה ההיררכי מוצג כרשומה, ולכן קיום הרשומה קובע למעשה את מבנה העץ. בתרשים 8—5 לדוגמא, ביטול רשומת העובד E1 מחייב ביטול של הרשומה P1 מבין רשומות הפרויקט כי אין קיום עצמאי לרשומת בן. ביטול צומת בעץ יכול לגרום לאבדן נתונים, כמו למשל העובדה שבמחלקה D1 היתה פעילות כלשהי בפרויקט P1. הדרך היחידה למנוע את אבדן הנתונים היא באמצעות רשומות ריקות.



תרשים 8 - 5: ייצוג צמתים בעץ על ידי רשימות

בשיטה השניה מוצגים הצמתים בעץ כרשומות מיוחדות בשעה שהרשומות האמיתיות מוצגות כצמתים סופיים בעץ ההיררכי. בתרשים 9—5 סומנו הרשומות המיוחדות בעיגול והרשומות האמיתיות סומנו במלבן. הרשומות המיוחדות מיועדות לשמירת המבנה ההיררכי

ואינן מכילות נתונים כלשהם. בשיטה זו ביטול רשומת העובד E1 אינה גורמת ביטול אוטומטי של הרשומה P1 ולא נגרם כאן אבדן נתונים. המבנה ההיררכי נשמר, ואינו תלוי בקיומה של רשומה כלשהי בבסיס הנתונים.



תרשים 9 – 5: ייצוג העץ ההיררכי באמצעות רשומות מיוחדות

מבחינת ייצוג המבנה ההיררכי שתי השיטות זהות. לשיטה השניה יש יתרונות ביכולתה לייצג את ההירכיה ללא תלות בקיום רשומות מסויימות. מאידך, השיטה השניה מסורבלת יותר הן מבחינת ההצגה הגרפית של המבנה ההיררכי והן מבחינת הטיפול בו. מערכת DL/1 של יבמ משתמשת בשיטה הראשונה, ומערכת SYSTEM 2000 של אינטל משתמשות בשיטה השנייה.

5. שפה לטיפול במבנה היררכי

המבנה ההיררכי מחייב שפה מיוחדת, **שפה היררכית** (Hierarchical Language), אשר מאפשרת את הקמת המבנה, ניהול הקשרים בין הרשומות וניהול הגישה לנתונים.

השפה ההיררכית מאפשרת **סריקת עץ** (Tree Traversal) בסדר מסויים. קיימות מספר שיטות שונות לסריקת מבני עץ אשר המקובלת ביניהן היא שיטת Pre-Order.

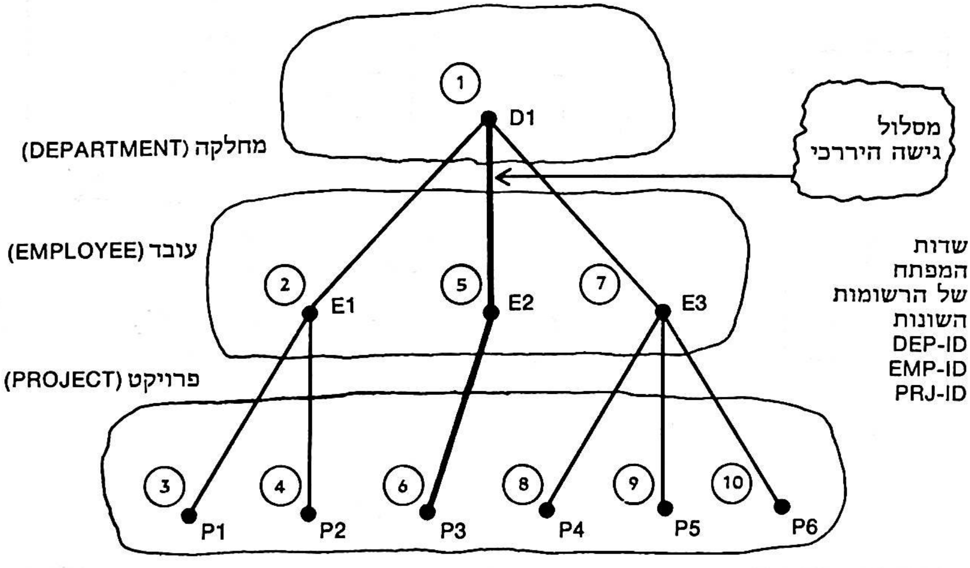
סדר הפעולות לסריקת העץ:

א. קרא את הרשומה, אם טרם קראת אותה,

ב. אחרת, קרא את כל רשומות הבן השמאליות שעדיין לא קראת,

ג. אחרת, אם אין יותר רשומות ברמות יותר נמוכות, חזור לרשומת האב,

ד. חזור לנקודת המוצא והמשך.



תרשים 10 - 5: סדר סריקת מבנה היררכי

כפי שניתן לראות בתרשים 10—5, סדר סריקת הרשומות לפי שיטת Preorder הינו מלמעלה למטה ומשמאל לימין. המסקנה היא ש"ביקור" ברשומות שמאליות מתבצע לפני הביקור בצמתים שמצד ימין. לעובדה אם הצומת הוא שמאלי או ימני יש **חשיבות** מבחינת מהירות הגישה אליו. כדי לבחור רשומה כלשהי מתוך המבנה ההיררכי יש להשתמש במסלול היררכי המוביל מרשומות השורש אל רשומת היעד. מסלול כזה מוצג בתרשים 10—5 בצורת קו מודגש D1—E2—P3.

כדי לאפשר את ביצוע הסריקה של המבנה ההיררכי מנהלת השפה **מראה־מקום** (Currency Indicator) המאפשר לתוכנית לעקוב אחר המיקום שלה במבנה ההיררכי.

5.1 פקודות של השפה ההירכית

הפקודות העיקריות של השפה ההירכית תוצגנה בדוגמאות המבוססות על תרשים 10—5. בפקודות דומות אך במבנה שונה משתמשים במערכת DL/1 של יבמ.

GET UNIQUE: פקודה זו מאפשרת **איתור רשומה בבסיס הנתונים** לפי תנאי מסויים. הרשומה הנקראת אינה תלויה במיקום התוכנית בבסיס הנתונים ולכן היא מיועדת לגישה ישירה לרשומת שורש או לרשומת בן כלשהי. בגמר ביצוע הפקודה מכיל מראה המקום את כתובת הרשומה הראשונה משמאל לרשומה שנשלפה (רשומה זו תהייה ברמה נמוכה יותר).

דוגמא לאיתור רשומה מחלקה עם מפתח D1:

```
GET  UNIQUE  DEPARTMENT
     WHERE   DEP-ID = D1.
```

שם סוג הרשומה (DEPARTMENT)

תנאי שליפה (DEP-ID = D1)

ניתן להשתמש בפקודה גם כדי לשלוף את רשומת העובד הראשונה במחלקה. בסיומה תוצב התוכנית על רשומה E1 שהיא הרשומה השמאלית ביותר לרשומת השורש D1.

```
GET  UNIQUE  EMPLOYEE
     WHERE   DEP-ID = D1.
```

ניתן להשתמש בפקודה זו כדי לשלוף את רשומת הפרויקט הראשונה של עובד E2 במחלקה D1. בסיומה תוצב התוכנית על רשומה P3.

```
GET  UNIQUE  PROJECT
     WHERE   DEP-ID = D1 AND
             EMP-ID = E2.
```

תנאי בוליאני (AND)

ניתן להשתמש בפקודה גם כדי לשלוף רשומת פרויקט מסויימת:

```
GET  UNIQUE  PROJECT
     WHERE   DEP-ID = D1 AND
             EMP-ID = E3 AND
             PRJ-ID = P6.
```

GET NEXT: פקודה זו מאפשרת **להתקדם במבנה ההירכי** לפי נוהל Preorder. ההתקדמות הינה מהמקום האחרון של התוכנית בבסיס הנתונים. לאחר שהתוכנית התמקמה על רשומת מחלקה D1, למשל, אפשר בעזרת פקודת GET NEXT לקרוא את רשומת העובד.

```
GET NEXT EMPLOYEE.
```

פקודה זו תקרא את הרשומה E1 למאגר. שימוש נוסף בפקודה יגרום את קריאת רשומה E2 ולאחר מכן את E3. שימוש בפקודה ללא ציון סוג הרשומה היה גורם את קריאת הרשומות לפי הסדר הבא: P6, P5, P4, E3, P3, E2, P2, P1, E1. ניתן להוסיף לפקודה תנאי, כדי לקרוא רשומות העונות על תנאי זה בלבד:

```
GET NEXT EMPLOYEE
      WHERE SALARY > 10000.
```

התנאי לאחזור הרשומה אינו מכיל את מפתח הרשומה אלא שדה כלשהו שלה. אם התוכנית ממוקמת על מחלקה D1 ואין אף עובד הממלא את התנאי הזה, הסריקה תעבור לעובדים של המחלקה הבאה, וכך הלאה, עד למציאת הרשומה הראשונה העונה על התנאי, או עד לסיום סריקת בסיס הנתונים. כדי למנוע את הגלישה יש להשתמש בפקודה הבאה.

GET NEXT WITHIN PARENT: פקודה זו דומה לפקודה GET NEXT אך היא מגבילה את התוכנית **לקרוא רק את רשומות הבן של רשומת האב** שעליה התוכנית ממוקמת. לדוגמא, לאחר שהתוכנית התמקמה על רשומת עובד E1, ניתן לקרוא את כל רשומות הפרוייקט על ידי שימוש חוזר בפקודה הבאה:

```
GET NEXT WITHIN PARENT PROJECT.
```

הפקודה תקרא את רשומות הפרויקט P1 ו־P2 ותאותת לתוכנית שאין רשומות פרויקט נוספת לעובד זה. שימוש בפקודת GET NEXT לא היה מסמן לתוכנית עובדה זו, והיא היתה עוברת לרשומת העובד הבא. אפשר להשתמש בתנאי לקריאת הרשומה, אך הכלל העיקרי ישמר — הבדיקה תבוצע ברשומות הבן השייכות לרשומת אב אחת.

```
GET NEXT WITHIN PARENT PROJECT
      WHERE DATE > 830801.
```

HOLD: לכל אחת מפקודות GET הקודמות ניתן להוסיף את הפרמטר HOLD לשם **נעילת הרשומה**. מערכת התוכנה אינה מאפשרת לתוכניות אחרות גישה אל הרשומה הנעולה כמו למשל, בעת עדכון או ביטול. היא מונעת פנייה בו־זמנית של תוכניות לאותה רשומה שלא למטרת קריאה בלבד, ובתוך כך — מונעת פגיעה בשלימות הנתונים.

```
GET UNIQUE HOLD EMPLOYEE
      WHERE DEP-ID = D1.
```

INSERT: הפקודה מאפשרת **הוספת רשומה חדשה** למבנה הירככי. לפני ביצוע הפקודה על התוכנית להגיע לרשומת האב בבסיס הנתונים באמצעות פקודות GET. לדוגמא, אם דרוש להוסיף רשומת פרויקט חדשה לעובד E2 יש תחילה להתמקם על רשומת העובד ("אב") ואחר כך להשתמש בפקודת INSERT. מערכת DBMS תקשור רשומה זו באופן אוטומטי לרשומת העובד.

REPLACE: הפקודה מאפשרת **עדכון תוכן השדות** המופיעים ברשומה. גם כאן יש תחילה להתמקם באמצעות פקודות GET מתאימות. קריאת הרשומה לשם עדכון חייב להתבצע בפקודת GET HOLD. לאחר שינוי תוכן השדות במאגר, תכתוב פקודת REPLACE את הרשומה המעודכנת בבסיס הנתונים.

DELETE: הפקודה **מבטלת את הרשומה** שהתוכנית ממוקמת עליה, ואת **כל** הרשומות התלויות בה. גם כאן יש להשתמש בפקודות GET, ולבסוף — לקרוא את הרשומה באמצעות הפקודה GET HOLD. השימוש בהוראה DELETE מחייב זהירות רבה מכיון שפקודת ביטול אחת עלולה לבטל את הרשומה ואת כל הרשומות התלויות בה (Triggered Delete). בפקודת DELETE הרגילה אפשר לבטל רשומת אב רק לאחר שבוטלו קודם לכן כל רשומות הבן שלה.

6. שאלות

1. נניח שקיימים שני עצים היררכיים. הגדר תת־סכימה המשתמשת ברשומות משני העצים. חקור את משמעות פעולות שפת הנתונים במובן של הפעולות האמיתיות שיש לבצע על בסיסי הנתונים האמיתיים.
2. נניח שקיים קשר N:M בין סטודנטים לקורסים. הצג את יישום הקשר הזה במערכת היררכית. איזו כפילו נתונים דרושה? תן מספר שאילתות לדוגמא.
3. האם העובדה שמחיקת רשומת במבנה ההירככי גורמת למחיקת כל רשומות הבן שלה היא יתרון או חסרון?
4. יש לנהל נתונים אודות מחלקות, פרויקטים, עובדים ומשימות. הקשרים בין הישויות הם.
   - מחלקה — עובד: חד—רב־ערכי
   - מחלקה—פרויקט: חד—רב־ערכי
   - עובד—פרויקט: רב—רב־ערכי
   - פרויקט—משימה: חד—רב־ערכי
   - עובד—משימה: רב—רב־ערכי

   א. הצג את המבנה ההירככי המטפל במבנה נתונים זה.

   ב. הראה כיצד תענה על השאילתות הבאות באמצעות שפת הנתונים ההירככית:
   - רשימת כל העובדים בפרויקט מסויים במחלקה.
   - רשימת המשימות שעובד מסויים ביצע, וכמות השעות שהושקעו על ידו.
   - רשימת המשימות לפרויקט מסויים וכמות השעות הכוללת שהושקעה בפרויקט.

פרק ו

מודל נתונים רשתי

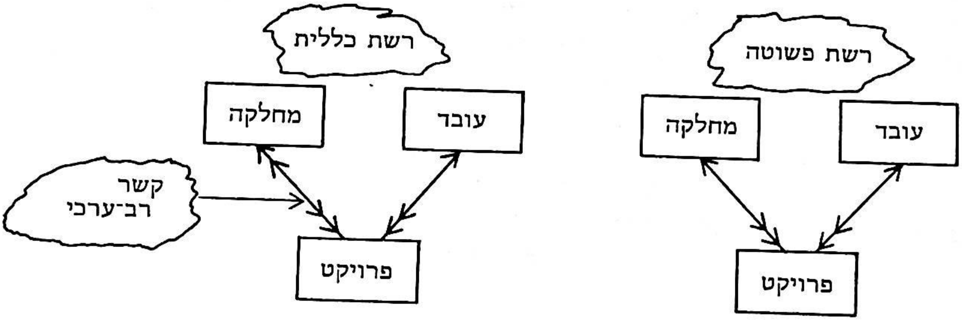
(Network Data Model)

1. מבוא
2. מודל הנתונים הרשתי - הגדרה
3. מושגי יסוד במודל הרשת
4. מבנים לוגיים המבוססים על חבורות
5. ייצוג קשרים N:M במודל נתונים רשתי
6. שטחי אחסון בבסיס הנתונים
7. שיטות לאיתור רשומות
8. זיהוי מופע של חבורה
9. שפה לטיפול בנתונים במודל רשתי
10. סיכום
11. שאלות

1. מבוא

מודל הנתונים הרשתי פותח כתוצאה מדרישה לטפל בקשרים מורכבים מאד בין הישויות השונות במודל הנתונים.

במודל הרשת הכללי מותרים כל סוגי הקשרים: חד—חד־ערכי, חד—רב־ערכי ורב—רב־ערכי. בגלל בעיות ביישום מבנה פיסי המייצג בצורה ישירה קשר רב—רב־ערכי. התפתח **מודל הרשת הפשוט** (Simple Network) שאינו מכיל קשרים רב—רב־ערכים. רוב מערכת DBMS המבוססות על מודל הרשת משתמשות ברשת פשוטה בלבד.



תרשים 1 – 6: רשת פשוטה ורשת כללית

מודל הרשת נוסח בצורה פורמלית בתחילת שנות ה־70 על־ידי צוות משימה (DBTG — Data Base Task Group) במסגרת ועידת CODASYL. הועידה — Conference On Data Systems Languages — היתה אחראית לפיתוח וקביעת תקנים לשפת COBOL. צוות המשימה שבו השתתפו אנשי אקדמיה, נציגי חברות מחשבים וחברות פרטיות עסק בקביעת תקנים לשפה מיוחדת להגדרת נתונים והשתמש למטרה זו במודל הרשת. החשיבות הרבה הנעוצה במודל הרשת שהומלץ על־ידי הצוות נובע מהעובדה שהוא אומץ על ידי קהילײת יצרני המחשבים ובתי התוכנה ושימש בסיס למספר מערכות DBMS מסחריות.

קיימות מספר רב של מערכות DBMS המבוססות על מודל זה, ביניהן IDMS של חברת Culinet עבור מחשבי IBM, DBMS-II של חברת DEC עבור מחשבי PDP וְ־VAX, IDS/II של חברת HONEYWELL, DMS 1100 של חברת UNIVAC ועוד.

בפרק זה נציג בצורה מפורטת את מודל הרשת כפי שהוגדר בהמלצות CODASYL.

2. מודל הנתונים הרשתי – הגדרה

מודל הנתונים הרשתי מבוסס על אוסף של סוגי רשומות ושל קשרים בין סוגי הרשומות. הרשומות מייצגות את הישויות ואת התכונות שלהן והקשרים בין סוגי הרשומות מייצגים את הקשרים שבין הישויות. ההגדרה הפורמלית של המודל הרשתי לפי המלצות CODASYL היא:

א. קיים אוסף של סוגי רשומות [R1, R2, R3, Rn]. לכל סוג רשומה מספר בלתי מוגבל של מופעים בבסיס הנתונים.

ב. קיים אוסף של קשרים, Lij, הקושרים את סוגי הרשומות למבנה נתונים רשתי אחד. הקשרים מקיימים את הכללים הבאים:

— הקשר Lij הינו פונקציונלי בכיוון אחד. לכל מופע של רשומה מסוג j יש בדיוק מופע אחד של רשומה מסוג i הקשור אליה. בכיוון השני הקשר אינו פונקציונלי, כלומר לכל מופע של רשומה מסוג i יש מספר מופעים של רשומות מסוג j הקשורים אליה. לסוג רשומה i נקרא **רשומת אב**, ולסוג רשומה j נקרא **רשומת בן**.

— בין כל שני סוגי רשומות Ri ו־Rj, יתכנו מספר קשרים שונים, הן מסוג Lij והן מסוג Lji. מכאן נובע שיש לקבוע לכל קשר שם אשר יזהה אותו בצורה חד־משמעית.

— ניתן להגדיר קשר מסוג Lii שהוא קשר בין מופעים שונים של סוגי רשומה זהים.

— ניתן להגדיר קשרים Lij ו־Lkj המצביעים על כך שלסוג רשומה j יש מספר רשומות אב שונות. תכונה זו מאפשרת את **ייצוג הרשת**.

3. מושגי יסוד במודל הרשת

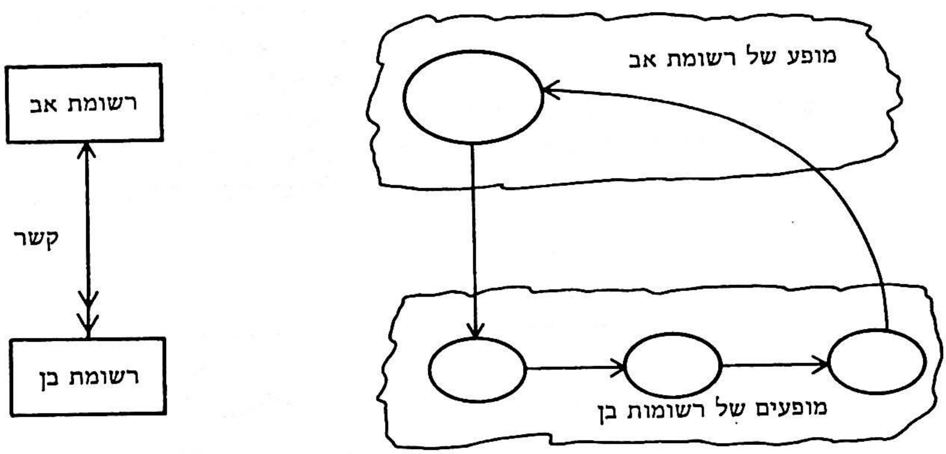
בהמלצות CODASYL מופיעים מונחים ומושגים שונים אשר נסביר בהמשך בצירוף דוגמאות.

רשומה (Record): הרשומה היא היחידה הבסיסית במודל הרשת. היא מכילה מספר קבוע או משתנה של תווים ומחולקת לשדות. אין אפשרות לפנות לשדה ללא שם הרשומה שבה הוא נמצא. הרשומה מייצגת את הישות, למשל "עובד".

סוג רשומה (Record Type): אוסף כל המופעים של רשומות במבנה זהה מוגדר כסוג רשומה. לדוגמא, סוג רשומה "מחלקה" כולל את **כל** המופעים של רשומות המחלקה אשר בבסיס הנתונים. סוג הרשומה מייצג את קבוצת הישות, למשל "מחלקה במפעל".

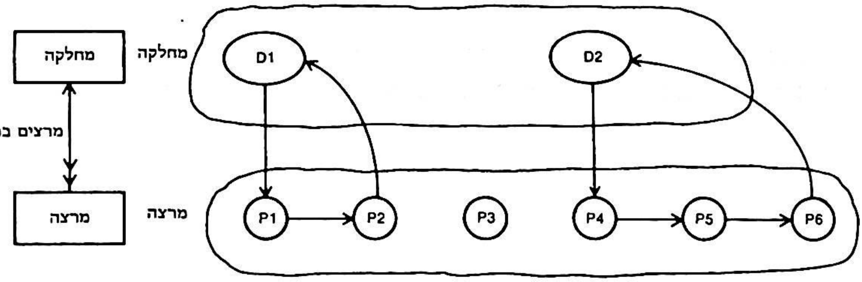
חבורה (Set): החבורה הינה קשר בין שני סוגי רשומה או יותר. החבורה במודל שונה לחלוטין מההגדרה המתמטית של קבוצה, למרות שהמונח האנגלי "Set" זהה. (מסיבה זו בחרתי להשתמש במונח "חבורה" ולא "קבוצה". כדי למנוע טעות נמצא גם בספרות המקצועית מונחים כגון DBTG Set או COSET—CODASYL Set במקום המונח Set). החבורה מגדירה סוג רשומה אחד **כרשומת אב** (Owner Record) ואחד או יותר סוגי רשומות המוגדרות **כרשומות בן** (Member Recrod).

בדומה לרשומות, גם לחבורה יש מופעים בבסיס הנתונים. מופע של חבורה כולל מופע אחד של רשומת אב, ומספר בלתי מוגבל של מופעי רשומות בן.



תרשים 2 – 6: ייצוג של חבורה במודל הנתונים הרשתי

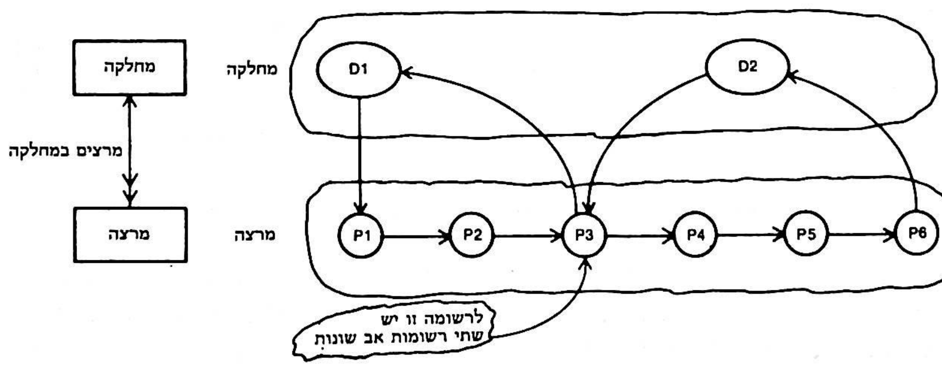
נשתמש בקשר בין רשומות מחלקה לרשומת מרצה כדי להדגים את מושג החבורה ואת מבני הנתונים שניתן לבנות מחבורות. תרשים 3—6 מציג שני מופעים של רשומות מחלקה, חמישה מופעים של רשומות מרצה ושני מופעים של החבורה "מרצים במחלקה" הקושרת את רשומת המחלקה עם רשומות המרצים השייכים למחלקה.



תרשים 3 – 6: דוגמא עם שני מופעים של חבורה

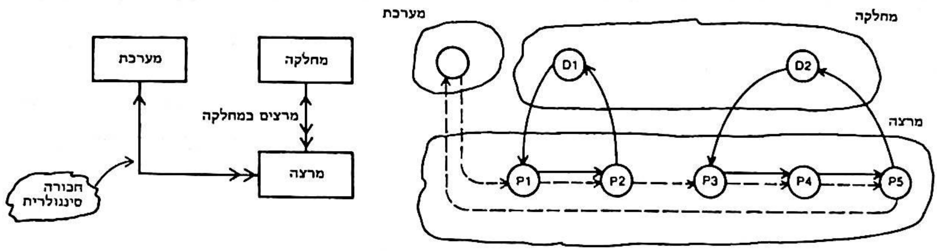
המודל אינו מונע קיום עצמאי של רשומות או של מופעים של סוגי רשומות. לפיכך אפשרי מופע של רשומה או של סוג רשומה שאינם קשורים לרשומות או סוגי רשומות אחרים. בתרשים 3—6 קיים מופע עצמאי של רשומת מרצה P3, שאינה קשורה לכל מופע של חבורה, למרות שבמודל מוגדת החבורה "מרצים במחלקה" הקושרת רשומות מרצה לרשומת מחלקה.

רשומת בן אינה יכולה להיות קשורה לשני מופעים שונים של חבורה, ולכן לכל רשומת בן יש רשומת אב אחת בלבד. אילוץ זה חשוב בזמן יישום מודל הרשת במערכות DBMS. בדוגמא שבתרשים 4—6 יש לרשומת מרצה P3 שתי רשומות אב שונות, D1 ו־D2 במסגרת החבורה "מרצים במחלקה". מודל הרשת מאפשר לשלוף את רשומת האב של כל רשומת בן. אך במקרה זה לא ניתן לזהות את רשומת האב הנכונה מכיון שרשומת הבן קשורה לשתי רשומות אב שונות. דבר זה נוגד את הגדרת המודל.



תרשים 4 – 6: רשומת בן הקשורה לשני מופעים שונים של חבורה

מודל הרשת מאפשר ייצוג של קבצים פשוטים באמצעות **החבורה הסינגולרית** (Singular Set). בחבורה זו רשומת האב מוגדרת כרשומה מיוחדת, **רשומת מערכת** (System Record) והיא קושרת אליה את **כל** המופעים של רשומות הבן. לכל חבורה סינגולרית יש מופע אחד בלבד בבסיס הנתונים.



תרשים 5 - 6: חבורה סינגולרית

החבורה הסינגולרית הינה **קובץ סדרתי פשוט** (Sequential) המכיל רשומות מאותו סוג. בתרשים 5—6 מוצגת חבורה סינגולרית הקושרת את כל רשומות המרצה לרשומת אב פיקטיבית אחת, רשומת המערכת, אשר מגדירה את "קובץ המרצים". סדר הרשומות בחבורה הסינגולרית אינו תלוי בסדר הרשומות בחבורה "מרצים במחלקה".

3.1 סדר החבורה (Set Order)

סדר החבורה קובע את מקום הוספת רשומת בן חדשה לחבורה. בהמלצות DBTG מופיעות מספר אפשרויות להוספת רשומה חדשה:

א. First — **הרשומה הראשונה** בחבורה, כלומר נוסיף אותה בתחילת השרשרת.
ב. Last — **הרשומה האחרונה** בחבורה, כלומר נוסיף אותה בסוף השרשרת.
ג. Next — **הרשומה הבאה** לאחר הרשומה שעליה ממוקמת תוכנית היישום. שיטה זו מאפשרת לתוכנית היישום לקבוע את מקום הוספת הרשומה החדשה לחבורה.
ד. Prior — **הרשומה הקודמת** לרשומה שעליה מוצבת תוכנית היישום.
ה. Sorted — **לפי סדר המיון** של שדה המפתח. הרשומות ממויינות לפי שדה מפתח כלשהו והרשומה תיכנס למקומה המתאים בשרשרת. המודל מאפשר לקבוע סדר מיון עולה או סדר מיון יורד, אם מותרים מפתחות כפולים או לא, והיכן יש להוסיף אותם כאשר הם מותרים.

מומלץ לא להשתמש באפשרויות Next ו־Prior מכיון שסדר החבורה גורר תלות בתוכנית היישום ועל־כן שינוי הסכימה עלול להשפיע על שלימות החבורה.

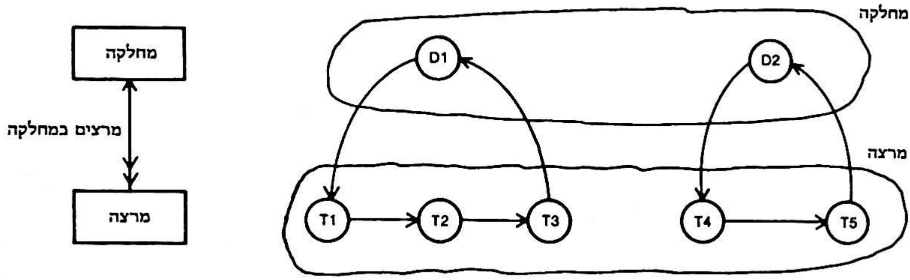
3.2 צורת הקשר של החבורה (Set linkage):

צורת הקשר בין רשומות הבן בחבורה מגדירה את מבנה החבורה: **מבנה שרשרת** או **רשומת מצביעים**.

שרשרת (Chain Set Mode)

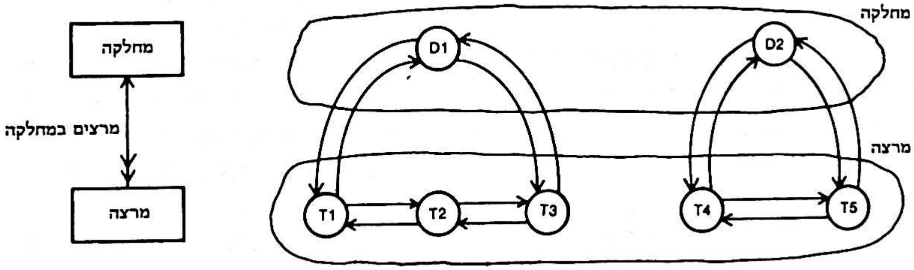
החבורה מיושמת באמצעות מבנה שבו כל רשומה מצביעה על הרשומה הבאה בקשר. דוח DBTG מציג שלש אפשרויות:

א. Next: הקשר מבוסס על מצביע לרשומה הבאה בחבורה. רשומת האב מצביעה לרשומת הבן הראשונה, רשומת הבן הראשונה מצביעה על השניה וכך הלאה. רשומת הבן האחרונה מצביעה חזרה לרשומת האב ובכך מתקבל מבנה של **שרשרת סגורה**, או **טבעת**. צורת קשר זו היא הכרחית ומופיעה בכל החבורות.



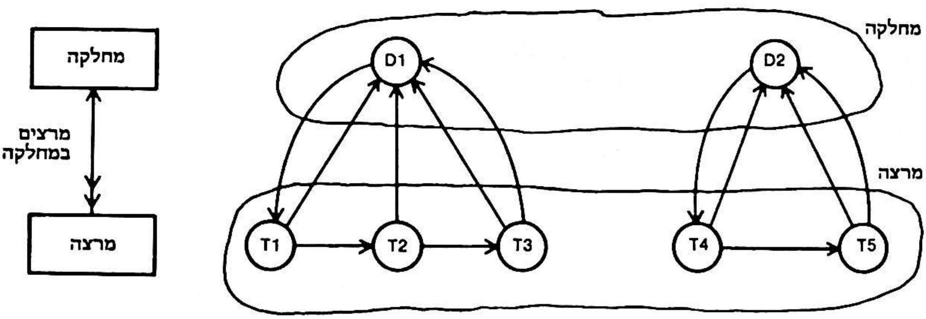
תרשים 6 – 6: חבורה עם מצביע Next בלבד

ב. Prior: הקשר מבוסס על מצביע מרשומת בן אחת לרשומת הבן הקודמת בחבורה ומרשומת הבן הראשונה אל רשומת האב. בשיטה זו קיימים שני מצביעים בכל חבורה, Next ו־Prior. קיום שני מצביעים אלה מאפשר את סריקת החבורה בשני כיוונים, לפנים ולאחור. הקשר הוא אופציונלי ונקבע בסכימה של המודל.



תרשים 7 – 6: חבורה עם מצביעים Next ו־Prior

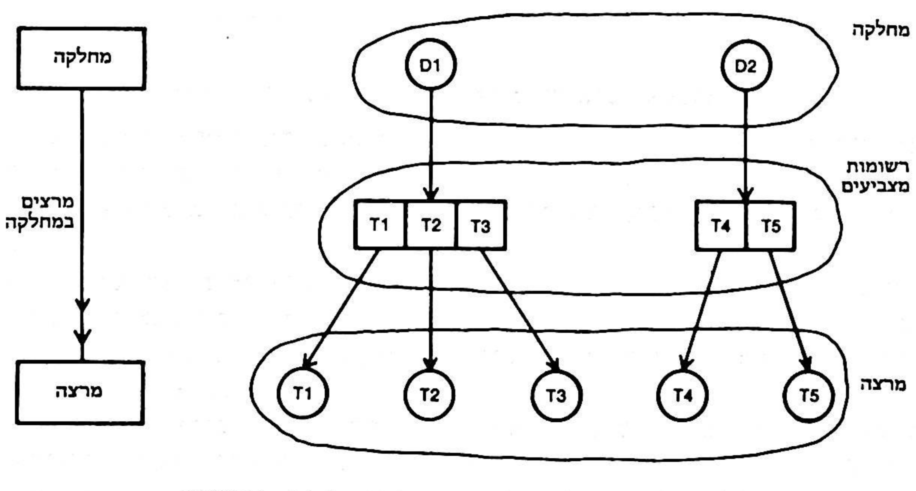
ג. **Owner**: כל רשומת בן בחבורה מצביעה על רשומת האב. מצביע זה מאפשר גישה מהירה מרשומת בן לרשומת אב והוא יעיל במיוחד אם מגיעים לרשומת הבן בחבורה אחת על סמך חברותה בחבורה אחרת. ניתן להגיע לרשומת האב גם תוך כדי שימוש במצביע Next, אבל שיטה זו מחייבת סריקה ושליפה של מספר רב של רשומות בן. צורת קשר זו היא אופציונלית ונקבעת בסכימה.



תרשים 8 – 6: חבורה עם מצביעים Next ו־Owner

רשומת מצביעים (Pointer Array):

מבנה שרשרת או טבעת מחייב סריקה סדרתית של רשומות הבן בחבורה. הסריקה עלולה להיות מאד לא יעילה כאשר רשומות הבן מפוזרות פיסית ביחידות האחסון. שיטה טובה יותר לביצוע גישה ישירה לרשומות הבן היא באמצעות רשומה נפרדת, שמכילה את המצביעים של החבורה, מצביע אחד לרשומת האב ומספר רב של מצביעים לרשומות הבן. בתרשים 9—6 מוצגות שתי רשומות מצביעים עבור המופעים של החברה "מרצים במחלקה" בהתאם לתרשים 8—6. לשיטה זו מספר חסרונות, במיוחד כאשר יש לעבור מחבורה לחבורה תוך כדי סריקה.



תרשים 9 – 6: קישור חבורה עם רשומת מצביעים

3.3 השתתפות בחבורה (Set Membership):

סוג החברות, או צירופים של סוגי חברות, קובעים את צורת השתתפות הרשומה בחבורה. סוג החברות קובע את זמן הוספת רשומת הבן לחבורה (בעת יצירת בסיס הנתונים או בכל זמן אחר) ואת אופי הקשר (קבוע או לא קבוע).

א. **חברות אוטומטית** (Automatic): כאשר מוסיפים רשומה לבסיס הנתונים מבצעת מערכת DBMS קישור מיידי (אוטומטי) שלה לאחד המופעים של החבורה. אם נגדיר את החבורה "מרצים במחלקה" כאוטומטית, יתבצע קישור של רשומת "מרצה" חדשה לאחד המופעים של החבורה באופן אוטומטי, מיד עם פתיחת רשומת המרצה. סוג חברות זה קובע שלא יתכן מרצה שאינו קשור למחלקה כלשהי, כלומר לא יתכן מופע עצמאי של רשומת "מרצה" בבסיס הנתונים.

— **חברות יזומה** (Manual): קישור הרשומה למופע כלשהו של החבורה מתבצע בצורה יזומה על־ידי המשתמש באמצעות פקודות מתאימות. אם נגדיר את החבורה "מרצים במחלקה" כיזומה, לא יתבצע קישור אוטומטי של רשומת "מרצה" חדשה לאחד המופעים בחבורה, מכיון שהקישור הוא בשליטת תוכנית היישום. סוג חברות זה מאפשר קיום רשומת "מרצה" עצמאית שאינה קשורה לשום מחלקה וניתן להוסיף אותה לבסיס הנתונים לפני שנקבע לאיזו מחלקה המרצה שייך. הקישור יבוצע אחר־כך בצורה יזומה.

— **חברות הכרחית** (Mandatory): לאחר קישור הרשומה למופע כלשהו של החבורה לא ניתן להוציאה מהחבורה למעמד עצמאי, אלא אם מבטלים אותה מבסיס הנתונים, אך ניתן להעביר את הרשומה למופעים אחרים של אותה חבורה. אם נגדיר את החבורה "מרצים במחלקה" כהכרחית, לא ניתן יהיה לנוציא את רשומת "מרצה" מהחבורה לאחר שבוצע הקישור שלה. לאחר שנקבעה למרצה מחלקה, הוא יכול לעבור למחלקה אחרת (מופע אחר של אותה חבורה) אבל אין אפשרות לקיים מופע עצמאי של רשומת "מרצה". בכל נקודת זמן הרשומה קשורה למופע כלשהו של החבורה.

— **חברות לא הכרחית** (Optional): לאחר קישור הרשומה למופע כלשהו של החבורה, ניתן להוציא את הרשומה מהחבורה מבלי שיהיה צורך לבטלה מבסיס הנתונים. אם נגדיר את החבורה "מרצים במחלקה" כלא הכרחית, ניתן יהיה להוציא את רשומת "מרצה" מהחבורה לאחר שבוצע הקישור שלה, ולהשאיר אותה בבסיס הנתונים כרשומה עצמאית.

סוג ההשתתפות בחבורה נקבע כצירוף של אחת מתוך ארבע אפשרויות:

— **אוטומטי־הכרחי**: קישור רשומה חדשה לחבורה מתבצע באופן אוטומטי כאשר מוסיפים אותה לבסיס הנתונים. לאחר שהרשומה נקשרה למופע כלשהו של החבורה לא ניתן לבטל את חברותה אלא אם מבטלים אותה מבסיס הנתונים. ניתן להעביר את הרשומה למופע אחר של החבורה.

— **אוטומטי־לא־הכרחי**: קישור רשומה חדשה לחבורה מתבצע באופן אוטומטי כאשר מוסיפים אותה לבסיס הנתונים. לאחר שהרשומה נקשרה למופע כלשהו של החבורה ניתן לבטל את חברותה בכל נקודת זמן. אין צורך לקשור אותה למופע אחר של אותה חבורה והיא יכולה להשאר באופן עצמאי.

— **יזום־הכרחי**: הוספת רשומה חדשה לבסיס הנתונים אינו מחייב קשירתה למופע כלשהו של החבורה. הקישור לחבורה יכול להתבצע בכל נקודת זמן, מאחר והוא נוצר על־ידי המשתמש באמצעות פקודות מיוחדות. לאחר שהרשומה נקשרה למופע כלשהו של החבורה לא

ניתן לבטל את חברותה, אלא אם מבטלים אותה מבסיס הנתונים. ניתן להעביר אותה למופע אחר של החבורה.

— **יזום־לא־הכרחי:** הוספת רשומה חדשה לבסיס הנתונים אינו מחייב קשירתה למופע כלשהו של החבורה. הקישור לחבורה יכול להתבצע בכל נקודת זמן על ידי תוכנית המשתמש באמצעות פקודות מיוחדות. לאחר שהרשומה נקשר למופע כלשהו של החבורה ניתן לבטל את חברותה בכל נקודת זמן. אין צורך לקשור אותה למופע אחר של אותה חבורה והיא יכולה להשאר עצמאית.

3.4 סיכום מאפייני החבורה

א. החבורה מייצגת קשר בין סוגי רשומות. לכל חבורה יש שם ייחודי והיא מכילה סוג רשומה אחד המוגדר כרשומת אב ואחד או יותר סוגי רשומות המוגדרים כרשומות בן.

ב. מופע של חבורה מורכב ממופע אחד של רשומת אב ומספר בלתי מוגבל של רשומות בן.

ג. במסגרת הסכימה ניתן להגדיר מספר בלתי מוגבל של חבורות.

ד. כל סוג רשומה יכול להיות רשומת אב בחבורה אחת או יותר. כל סוג רשומה יכול להיות רשומת בן באחת או יותר חבורות. אותו סוג רשומה יכול להיות רשומת אב באחת או יותר חבורות ורשומת בן באחת או יותר חבורות אחרות.

ה. אותו סוג רשומה אינו יכול להיות גם רשומת אב וגם רשומת בן באותה חבורה. המודל מרשה רשומת אב אחת לכל רשומת בן באותה חבורה.

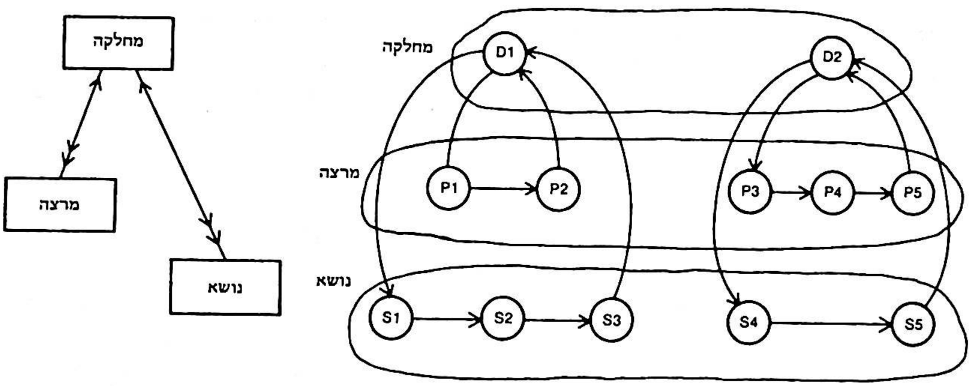
ו. בכל חבורה יש להגדיר את סדר רשומות הבן באחת מהאפשרויות הקיימות. לא ניתן להגדיר סדר לרשומות אב פרט למקרה שהן מוצגות כרשומות בן בחבורה אחרת.

ז. חבורה סינגולרית היא חבורה מיוחדת שבה קיימת רשומת אב אחת בלבד, רשומת מערכת.

4. מבנים לוגיים המבוססים על חבורות

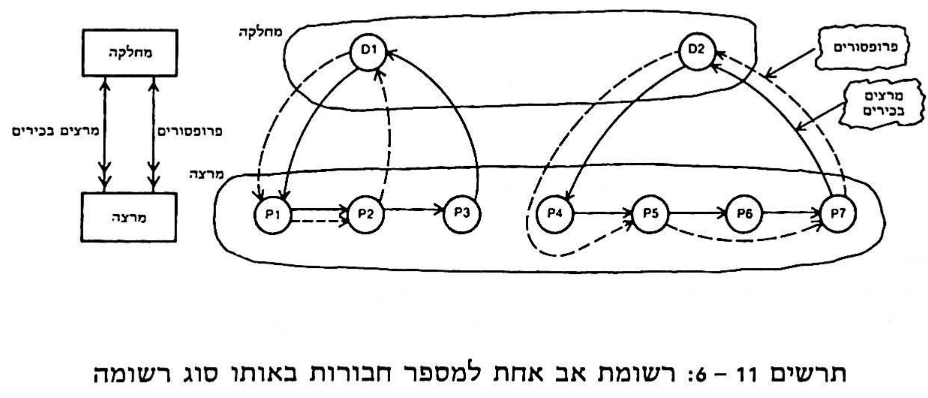
החבורה הינה מבנה היררכי בעל שתי רמות. מודל הרשת משתמש במבנה בסיסי זה כאבן בנין למבני נתונים מורכבים ביותר. בסעיף זה נציג את הכללים הבסיסיים בהרכבת מבני נתונים במודל הרשתי.

Multiple Ownership: סוג רשומה כלשהו יכול לשמש כרשומת אב במספר חבורות שונות. תרשים 10—6 מציג דוגמא שבה סוג רשומה "מחלקה" הוא רשומת אב בשתי חבורות שונות: "מרצים במחלקה", "נושאים נלמדים".



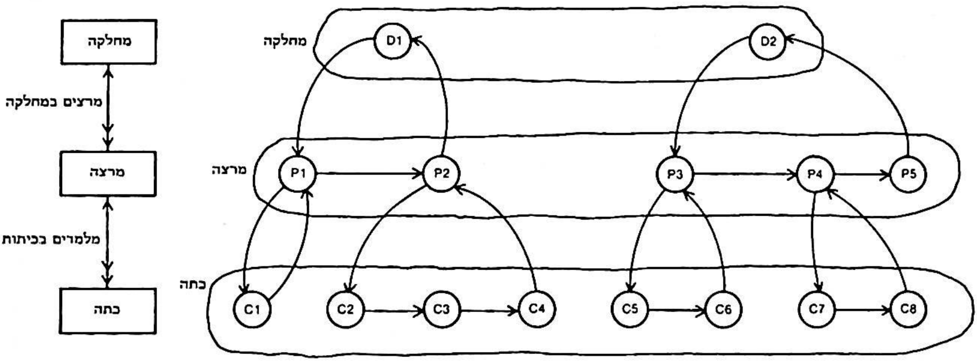
תרשים 10 - 6: רשומת אב אחת למספר חבורות בסוגי רשומה שונים

מודל הרשת מאפשר להגדיר מספר קשרים שונים בין שני סוגי רשומות. בתרשים 11—6 סוג רשומה "מחלקה" הוא רשומת אב בשתי חבורות שונות "פרופסורים" ו"מרצים בכירים" הקושרות שני סוגי רשומה בלבד. במחלקה D1 לדוגמא, רק המרצים P1 וְ־P2 הם בדרגת פרופסור ואילו במחלקה D2 רק המרצים P5 וְ־P7 הם בדרגה זו.



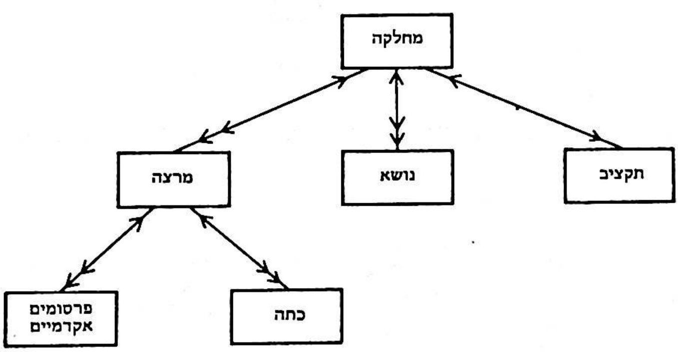
תרשים 11 - 6: רשומת אב אחת למספר חבורות באותו סוג רשומה

Membership and Ownership: סוג רשומה אחד יכול להיות רשומת בן בחבורה אחת ורשומת אב בחבורה אחרת. תכונה זו מאפשרת **ייצוג מבנים היררכיים** במספר רמות כלשהו. בתרשים 6—12 סוג רשומה "מרצה" משתתף כרשומת בן בחבורה "מרצים במחלקה" ובאותה עת הוא גם רשומת אב בחבורה "מלמדים בכתות".



תרשים 12 - 6: סוג רשומה אחד בתפקידים שונים

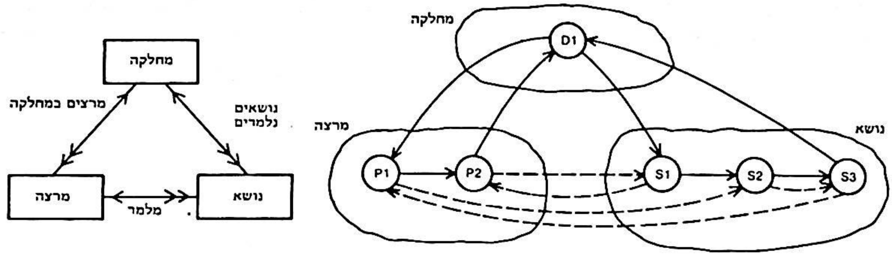
יצוג של מבנה הירכי באמצעות מודל הרשת מתאפשר על ידי שילוב של שתי תכונות: אפשרות כל סוג רשומה להיות רשומת אב במספר חבורות והאפשרות של כל סוג רשומה להיות רשומת אב ורשומת בן בעת ובעונה אחת. המודל ההירכי מוצג במודל הרשת כאוסף של חבורות המקיימות את דרישות המודל ההירכי.



תרשים 13 - 6: מבנה הירכי כאוסף של חבורות

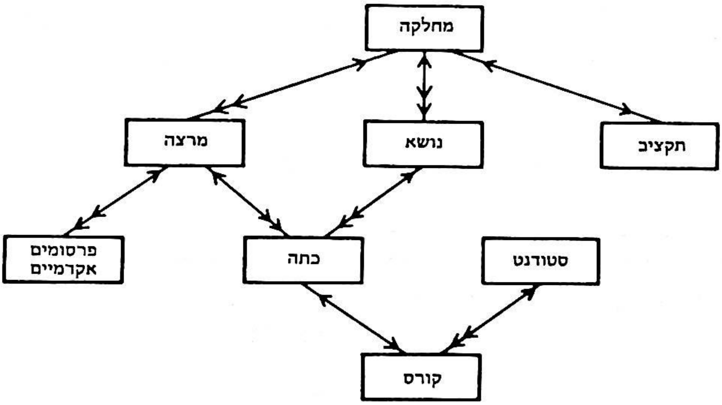
Multiple Membership: סוג רשומה אחד יכול להיות רשומת בן במספר חבורות שונות. תכונה זו מאפשרת את ייצוג הרשתות במודל הנתונים הרשתי.

תרשים 6—14 מציג דוגמא שבה סוג רשומה "נושא" הינו רשומת בן בחבורה "נושאים נלמדים" וגם בחבורה "מלמד". מרצה P1, למשל, שייך למחלקה D1 ומלמד נושאים S2 ו־S3



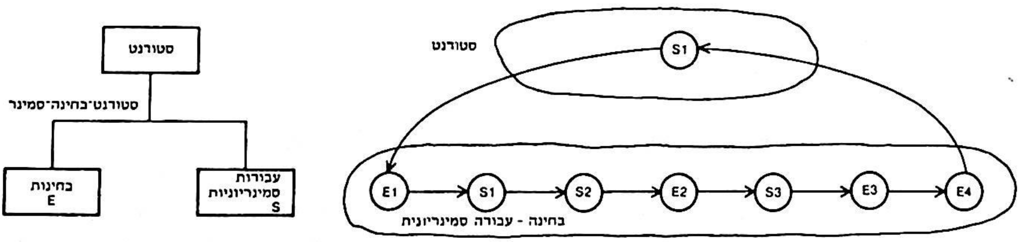
תרשים 14 – 6: רשומת בן למספר חבורות באותו סוג רשומה

השילוב בין שלש התכונות שהוצגו עד כה מאפשר תיאור מבנה רשתי מורכב בדומה לזה שבתרשים 15—6.



תרשים 15 – 6: מבנה רשתי

Multimember: בחבורה מסוג זה יש סוגי רשומות בן שונים לרשומת אב אחת.

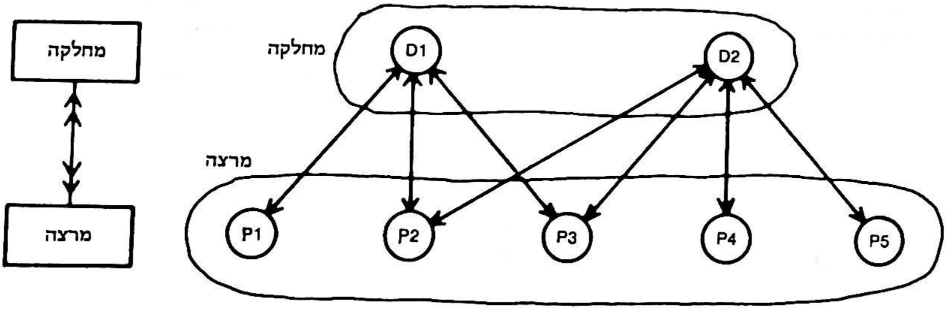


תרשים 16 – 6: חבורה עם סוגים שונים של רשומות בן

5. ייצוג קשרים N:M במודל נתונים רשתי

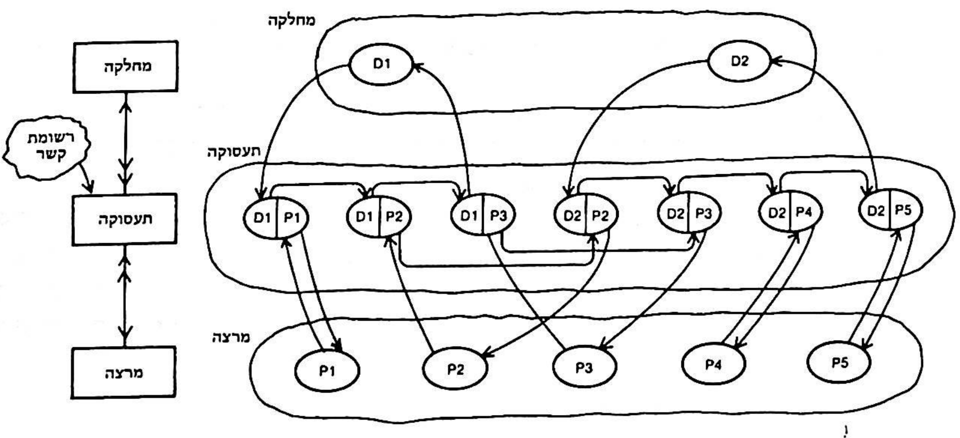
בדומה למודל הנתונים ההירכי גם המודל הרשתי אינו מאפשר ייצוג ישיר של קשרים מסוג N:M. אפשר לתאר קשרים אלה על־ידי פירוק הקשר הרב—רב־ערכי לשני קשרים חד—רב־ערכיים ושימוש בסוג רשומה נוסף, **רשומת הקשר** (Intersection Record).

בתרשים 17—6 מתוארת רשת כללית אשר מכילה קשר רב—רב־ערכי בין רשומת "מחלקה" ורשומת "מרצה". כפי שניתן לראות המרצים P2 ו־P3 מלמדים בשתי מחלקות שונות.



תרשים 17 - 6: רשת כללית

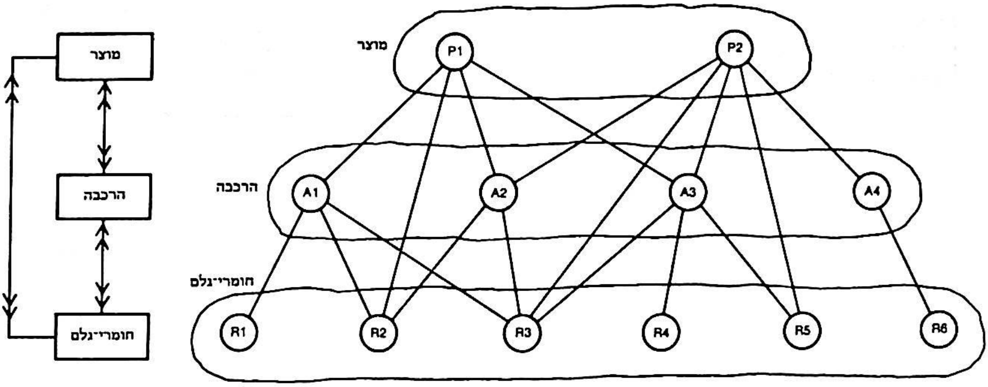
הקשר שמוצג בתרשים 17—6 הינו מסוג N:M אשר אינו מוכר במודל DBTG. כדי לתאר אותו משתמשים ברשומת הקשר "תעסוקה" אשר נמצאת בין רשומת "מחלקה" לבין רשומת "מרצה". הפירוק לשני קשרים חד—רב־ערכיים ניתן בתרשים 18—6.



תרשים 18 - 6: שימוש ברשומת קשר לייצוג קשר רב - רב־ערכי

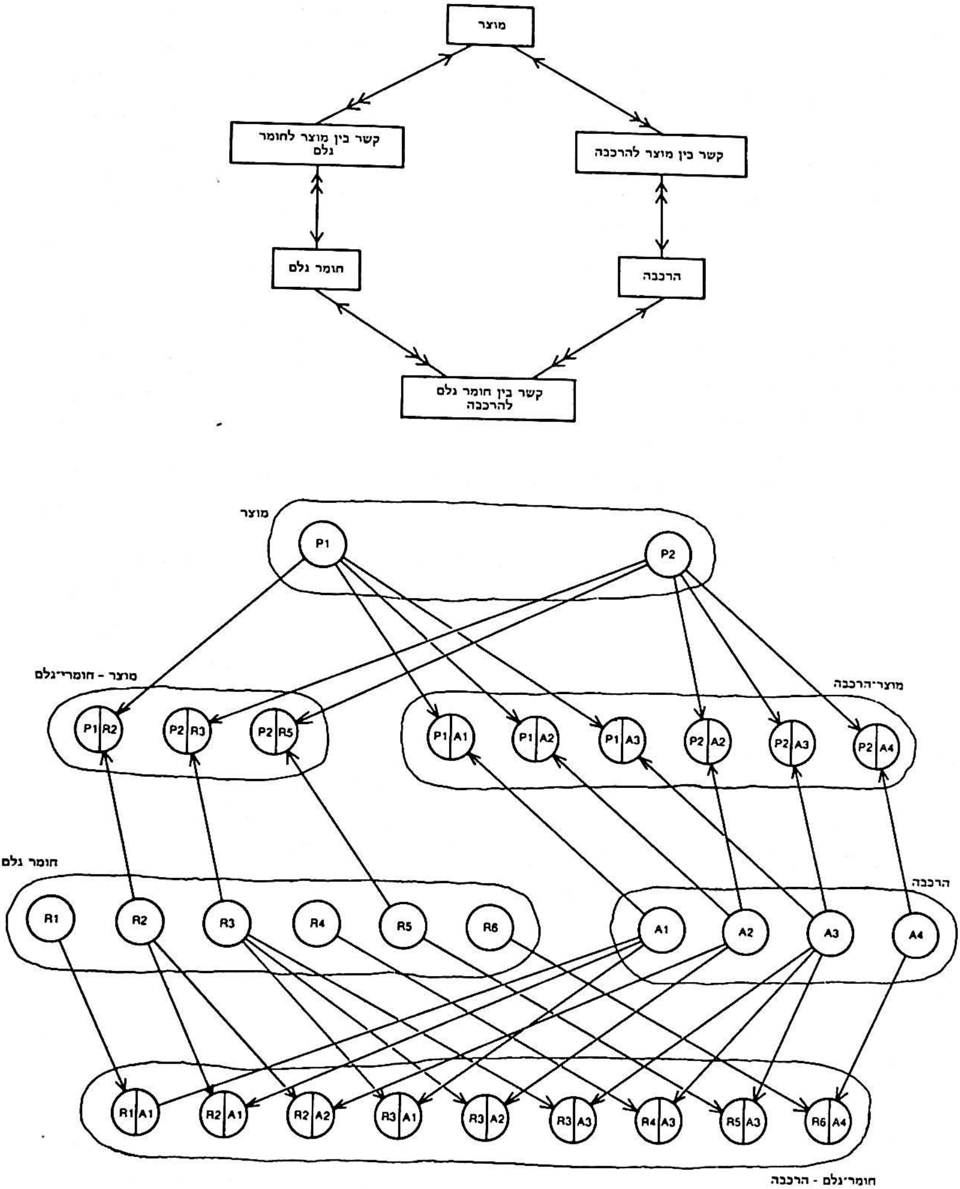
אפשר להשתמש בשיטה זו על פי התכונה Multiple Membership של רשומה במודל DBTG שמגדירה לרשומת "תעסוקה" שתי רשומות אב, רשומת "מחלקה" ורשומת "מרצה". רשומת הקשר אינה מיועדת לייצוג קשרים רב—רב־ערכיים בלבד. היא מכילה בדרך כלל **נתוני קשר** (Intersection Data) משמעותיים עבור שתי הרשומות. בדוגמא זו ניתן לשמור ברשומת "תעסוקה" נתונים כגון "תאריך תחילת עבודת המרצה במחלקה", "מספר שעות שבועיות שהמרצה מלמד במחלקה" ועוד.

דוגמא נוספת לשימוש ברשומת קשר ניתנת בתרשים 6—19 המתאר מבנה עץ מוצר כללי. **מבנה המוצר** (BOM-Bill of Material) מכיל צירוף של חומרי גלם והרכבות אשר יכולות גם הן להכיל צירוף של הרכבות משנה וחומרי גלם עד לרמה הנמוכה ביותר של ההרכבה אשר מכילה צירוף של חומרי גלם בלבד. בין שלושת סוגי הרשומה "מוצרים", "הרכבות" ו"חומרי גלם" קיימים קשרים רב—רב־ערכיים.



תרשים 19 - 6: מבנה עץ מוצר כללי

לתיאור עץ מוצר במודל הרשת יש להשתמש ברשומות קשר ובנתוני קשר כפי שהוסבר קודם. בעץ המוצר קיימים נתונים חשובים על המבנה כמו למשל "כמות בה מופיע אלמנט הבן באלמנט האב", או פשוט יותר: כמה גלגלים יש במכונית אחת. כך למשל, בנוסף לידיעה שמוצר P1 מכיל את ההרכבות A1, A2, A3 וחומר גלם R2 יש לדעת למשל כמה יחידות A1 נמצאות ב־P1. המקום הטבעי לרשום נתונים אלו הוא ברשומת הקשר. תרשים 6—20 מתאר עץ מוצר כללי ורשומות קשר. קיימות טכניקות נוספות לייצוג עצי מוצר במודל נתונים רשתי ולא נדון בהן כאן.



תרשים 20 – 6: ייצוג עץ מוצר במודל נתונים רשתי

6. שטחי אחסון בבסיס הנתונים

בהמלצות CODASYL כלול המושג **שטח אחסון** (Area) במסגרת הסכימה המגדירה את המבנה הלוגי של בסיס הנתונים. הגדרת שטח האחסון במסגרת המבנה הלוגי ולא במסגרת המבנה הפיסי נותנת בידי המתכנן ומנהל בסיס הנתונים כלים מסויימים לאופטימיזציה של הגישות לבסיס הנתונים.

שטח האחסון הינו תת־חלוקה פיסית של השטח שבו מאוחסן בסיס הנתונים אשר מאפשרת למערכת ההפעלה להתייחס אל השטח כאל קובץ. המודל מאפשר את חלוקת בסיס הנתונים למספר בלתי מוגבל של שטחי אחסון אשר לכל אחד מהם נקבע שם נפרד במסגרת הסכימה, לדוגמא:

AREA NAME IS DEPT-AREA

המודל מאפשר לקבוע במסגרת הסכימה אילו סוגי רשומות יאוחסנו בכל שטח אחסון, ואם יהיה זה סוג רשומה אחד או סוגי רשומות שונים. למרות שאפשר לאחסן את כל סוגי הרשומות בשטח אחד, מעדיפים את הפרדתם לשטחי אחסון שונים כדי להשיג יעילות גדולה יותר בגישות לבסיס הנתונים. לאחר שנקבע השיוך של סוגי הרשומה לשטח האחסון אין אפשרות להעביר אותה לשטח אחסון אחר, אלא אם טוענים מחדש את בסיס הנתונים.

הקצאת סוג רשומה DEPARTMENT לשטח אחסון DEPT-AREA תיעשה בצורה הבאה:

DEPARTMENT RECORD WITHIN DEPT-AREA

המודל מאפשר אחסון רשומת בן (TEACHER) באותו שטח שבו מאוחסנת רשומת האב שלה (OWNER),

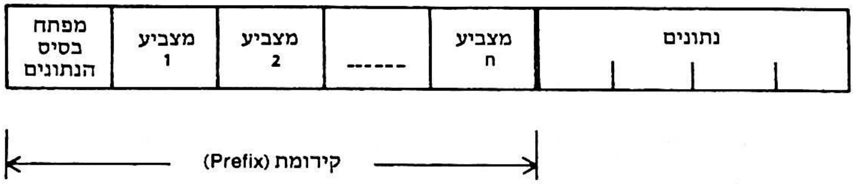
לדוגמא: TEACHER RECORD WITHIN AREA OF OWNER

השטח עצמו מחולק **לדפים** (Pages) והדף מחולק **לשורות** (Rows). **המופע** (Occurrence) של הרשומה מאוחסן בשורה. המשתמש יכול לקבוע את גודל השטח על־ידי ציון מספר הדפים בשטח האחסון.

למרות שמושג השטח מתייחס למבנה הפיסי הוא אינו מבחין בתכונות הפיסיות של יחידת האחסון. כתוצאה מכך מושגת פשרה בין הרצון ליצור אי־תלות פיסית לבין הרצון לקבל ביצועים טובים ממערכת ניהול בסיס הנתונים.

7. שיטות לאיתור רשומות

השיטה לאיתור רשומה (Record Location Mode) קובעת את הצורה שבה מערכת DBMS תאחסן ותשלוף את הרשומה. לכל סוג רשומה יש שיטת איתור קבועה אשר מוגדרת בסכימה. הרשומות מאוחסנות בשטחי אחסון שונים וכל מופע שלהן מקבל זיהוי חד־משמעי בבסיס הנתונים, אשר מורכב ממספר הדף ומספר השורה בדף שבה מאוחסנת הרשומה. קוד זיהוי מופע הרשומה קרוי **מפתח בסיס הנתונים** (DB-Key — Data Base Key). מפתחות אלה הם **מצביעים** (Pointers) בחבורות השונות שבהן הרשומה משתתפת.



תרשים 21 - 6: מבנה רשומה בבסיס הנתונים

ברשומה לדוגמא שבתרשים 21—6 ישנם שני חלקים עיקריים:

— **קידומת** המכילה את מפתח בסיס הנתונים ואת כל המצביעים הדרושים לקישור הרשומה לכל החבורות שבהן היא משתתפת, אם כרשומת אב ואם כרשומת בן,

— **נתונים** המחולקים לשדות. המשתמש יכול להגדיר שדה אחד כמפתח לוגי של הרשומה ושדות נוספים שיהיו שדות מיון עבור המופעים השונים בחבורה.

קיימות שלשת שיטות לאחסון ולאיתור רשומה בבסיס הנתונים:

א. **ישירה** (DIRECT): הרשומה מאוחסנת בדף ובשורה אשר נקבעים על־ידי המשתמש באמצעות מפתח בסיס הנתונים. קיימות פקודות המאפשרות קביעה ושינוי של המפתחות, וכן כתיבה וקריאה של רשומות על פי מפתחות אלו. לא מומלץ להשתמש בשיטה זו.

ב. **חישובית** (CALC): אחד משדות הנתונים מוגדר כמפתח לוגי ועליו מפעילה מערכת DBMS תהליך חישובי (Hashing) שהופך אותו למפתח בסיס הנתונים. שיטה זו מאפשרת גישה ישירה לרשומה על־פי ידיעת המפתח הלוגי שלה כמו תעודת זהות למשל.

ג. **עקיפה** (VIA): שיטת איתור זו מיועדת לאיתור רשומות בן בחבורה כלשהי כאשר אין צורך בגישה ישירה אלא רק בגישה המסתמכת על חברות הרשומה בחבורה. בשיטה זו המופעים של רשומות הבן יאוחסנו באותה סביבה פיסית (דף למשל) של רשומת האב. אם אין מקום באותה סביבה, מערכת DBMS תנסה לאתר מקום פנוי בסביבה קרובה אחרת. שיטה זו מגדילה את יעילות הביצוע מכיון שניתן לקבל את כל הרשומות השייכות למופע אחד של החבורה בפעולת קלט/פלט אחת. כאשר החבורה גדולה מאד עלולה קריאת כל החבורה להיות מגבלה. אם הרשומה היא רשומת בן במספר חבורות שונות על המשתמש לקבוע באיזו מהן יוגדר הקשר VIA. הוא יבחר למטרה זו בחבורה שתדירות הגישה אליה היא הגבוהה ביותר.

8. זיהוי חבורה

פעולות מסויימות בבסיס הנתונים מחייבות **זיהוי החבורה** (Set Occurrence Selection) שבה יש לבצע אותן. למשל, כאשר יש להוסיף רשומה לבסיס הנתונים יש לאתר את החבורה ואת המקום בתוכה שאליו יש לקשור אותה. זיהוי החבורה יכול להעשות בדרכים שונות.

א. זיהוי החבורה על־ידי התוכנית

תוכנית היישום מגיעה למקום בבסיס הנתונים שבו יש להוסיף את הרשומה החדשה. שיטת זיהוי זו מוגדרת בסכימה באמצעות המשפט הבא:

SET OCCURRENCE SELECTION IS THRU CURRENT OF SET.

אם סוג ההשתתפות בחבורה הוא **יזום** (Manual) תוכל תוכנית היישום להוסיף רשומה חדשה לבסיס הנתונים לאחר איתור המופע המתאים של החבורה ולקשור את הרשומה למופע שנבחר. שיטת הזיהוי על־ידי התוכנית מחייבת את תוכנית היישום להכיר את מבנה בסיס הנתונים ולנווט בתוכו. שיטות הזיהוי האחרות אינן תלויות כל כך בהכרת מבנה מסיס הנתונים.

ב. זיהוי החבורה על־ידי שגרה

הזיהוי מתבצע על ידי שגרה כתובה מראש ומאוחסנת בספריית השגרות של מערכת DBMS. שיטת זיהוי זו מוגדרת בסכימה באמצעות המשפט הבא שבו ABC הוא שם השגרה:

SET OCCURRENCE SELECTION IS BY PROCEDURE ABC.

ג. זיהוי רשומת האב של החבורה

מערכת DBMS מאתרת את המופע של רשומת האב של החבורה ולאחר מכן מוסיפה את הרשומה החדשה. שיטת זיהוי זו מוגדרת בסכימה באמצעות המשפט הבא שבו data-item i הוא שם שדה (אחד או יותר) שבו מוסרת תוכנית היישום את גורם החיפוש, כפי שיוסבר בהמשך:

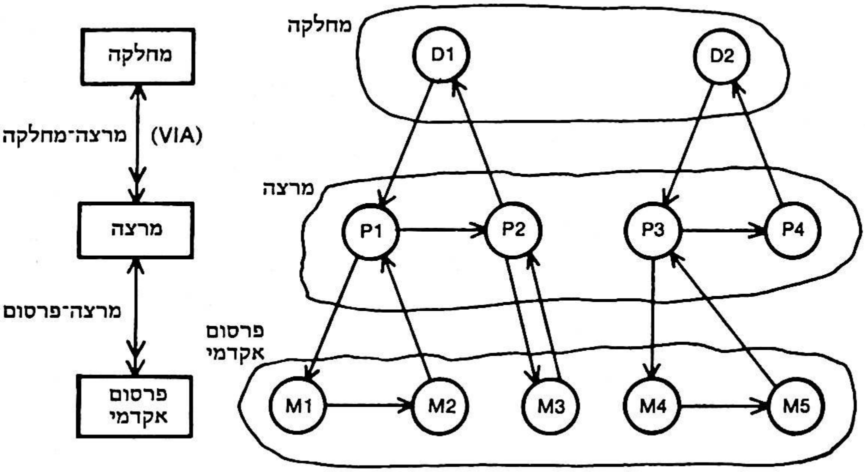
SET OCCURRENCE SELECTION IS THRU LOCATION MODE
OF OWNER USING data-item1, data-item2,...

בכל אחת משיטות האיתור מוטלים על תוכנית היישום תפקידים שונים:

א. בשיטת איתור CALC של רשומת אב על תוכנית היישום לספק למערכת DBMS את מפתחות החישוב המתאימים. נשתמש בדוגמא שבתרשים 19—6: כדי להוסיף רשומת "מרצה" יש לזהות את רשומת המחלקה שבה הוא מועסק. המשתמש צריך לקבוע את תוכן מפתח המחלקה, למשל "מספר המחלקה", ומערכת DBMS תאתר באופן אוטומטי את רשומת המחלקה ותקשר את רשומת המרצה למופע של החבורה. אין תוכנית היישום צריכה לאתר את רשומת האב כי כל הפעולה נעשית על ידי מערכת התוכנה בלבד.

ב. בשיטת האיתור DIRECT של רשומת האב על תוכנית היישום לספק למערכת DBMS את מפתח־בסיס־הנתונים המתאים. אין התוכנית היישום צריכה לאתר את רשומת האב.

ג. בשיטת האיתור VIA של רשומת האב על תוכנית היישום לספק למערכת DBMS נתונים על המופע של רשומת האב שיש לבחור. מערכת DBMS חייבת לזהות את רשומת האב על פי חברותה בחבורה אחרת. והדבר מחייב לזהות גם את המופע של החבורה השניה בהתאם לשיטת האיתור של רשומת האב. ניתן לחזור על מספר רמות של זיהוי רשומת אב בתנאי שהזיהוי יסתיים עם רשומת אב ששיטת האיתור שלה היא CALC או DIRECT. רשומה זו מזהה מסלול ישיר אל רשומת האב הדרושה. נשתמש בדוגמא להסבר.



תרשים 22 – 6: איתור רשומת אב בשיטת VIA

בדוגמא המתוארת בתרשים 22—6 רשומת־אב של חבורה "מרצה־מחלקה" קושרת את רשומת "מרצה" עם רשומת "מחלקה" בקשר VIA. כדי להוסיף רשומת "פרסום אקדמי" לחבורה "פרסומים אקדמיים" יש לזהות את רשומת המרצה וגם את רשומת המחלקה. מערכת DBMS אינה יכולה לאתר את המחלקה שבה מועסק המרצה ולכן על תוכנית היישום לספק לה את מפתח המחלקה יחד עם מפתח המרצה. המערכת תאתר את רשומת המחלקה בהתאם לשיטת האיתור שלה, תסרוק את שרשרת רשומות המרצה עד למציאת הרשומה המבוקשת ותוסיף את רשומת הפרסום האקדמי לחבורה המתאימה. בחבורה שבה הקשר הוא VIA נדרש שרשומת האב תהייה רשומת בן ממויינת.

היתרון של שיטת זיהוי החבורה באמצעות זיהוי רשומת האב הוא בכך שתוכנית היישום אינה צריכה להכיר את מבנה בסיס הנתונים ולנווט בתוכו עד לאיתור המקום להוספת הרשומה החדשה. התוכנית צריכה לספק את המפתחות המתאימים ומערכת DBMS מטפלת בהוספת הרשומה למקומה הנכון בבסיס הנתונים. שיטה זו מפשטת במידת מה את תוכנית היישום.

9. שפה לטיפול בנתונים

בנוסף להגדרת מבנה הנתונים המליץ דוח DBTG גם על מבנה שפה לטיפול בנתונים. תוכניות היישום כתובות בשפה מארחת כמו COBOL אשר מכילה אוסף של פקודות DML מיוחדות המאפשרות את הטיפול בנתונים.

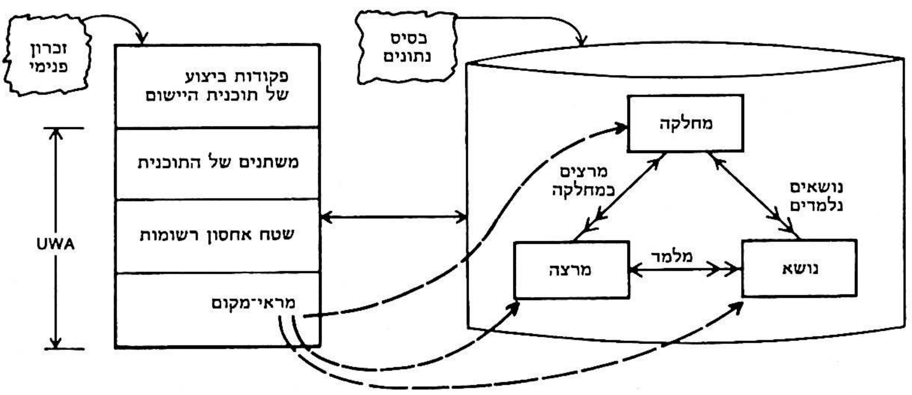
9.1 שטח עבודה של התוכנית

תוכנית היישום מחולקת באופן כללי לשניים. איזור בזכרון המכיל את **פקודות הביצוע** ואיזור המוגדר **כשטח עבודה** של התוכנית (UWA—User Working Area). שטח העבודה מחולק לשלשה חלקים, כמודגם בתרשים 23—6:

א. **משתנים** של תוכנית היישום, כמקובל בכל תוכנית.

ב. **שטח אחסון רשומות** מכיל את תוכן הרשומות השונות שנקראו או עומדות להכתב לבסיס הנתונים. בשטח האחסון יש מקום נפרד לכל סוג רשומה המוגדר בתת־הסכימה ויש בו גם מקום להעברת פרמטרים לפקודות DML שונות. למשל, לקריאת רשומה מבסיס הנתונים יש להעביר לתוך שטח האחסון של סוג הרשומה את המפתח של הרשומה המבוקשת.

ג. **מראי־מקום** (Currency Indicators) הם אחד המרכיבים החשובים של שפת DML. מראי המקום מציינים את מיקום תוכנית היישום בבסיס הנתונים ומאפשרים את הניווט (Navigation) שלה בתוך מבנה הרשת.



תרשים 23 - 6: מבנה תוכנית יישום

הצורך במראי מקום נובע מהעובדה שבמבנה הרשתי קיימים **מספר מסלולים שונים לניווט** בין רשומת מקור לרשומת יעד כלשהי. בתרשים 23—6 מוצג מבנה רשת שמכיל שלש חבורות שונות. כדי להגיע מרשומת "מחלקה" לרשומת "נושא" ניתן לעבור דרך רשומות "מרצה" או להגיע ישירות מרשומת "מחלקה".

במודל הנתונים ההיררכי קיים **מסלול אחד בלבד** בין כל רשומת מקור לכל רשומת יעד ולכן אין צורך במנגנון מראי מקום. במודל הרשתי לא ניתן להגדיר מסלול יחיד בין שני סוגי

רשומות ועל כן נדרשת הטכניקה של מראי־מקום.

דוח DBTG ממליץ על מספר מראי־מקום שיאפשרו את הניווט בבסיס הנתונים:

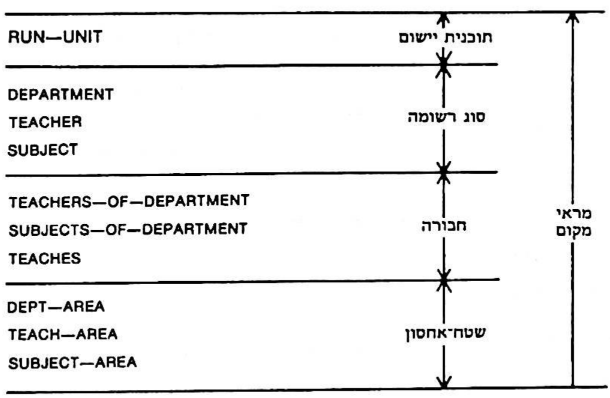
א. **מראה מקום לסוג רשומה** (Current of Record Type) מורה על הרשומה האחרונה שתוכנית היישום טיפלה בה. לכל סוג רשומה שבתת־סכימה של תוכנית היישום יש מראה מקום אחד מסוג זה.

ב. **מראה מקום לחבורה** (Current of Set Type) מצביע על המופע האחרון של רשומה בחבורה שטופלה על־ידי תוכנית היישום. לעיתים ההצבעה תהייה על מופע של רשומת אב ולעיתים — על מופע של רשומת בן. לכל חבורה שמופיעה בתת־הסכימה של תוכנית היישום יש מראה מקום אחד מסוג זה. השימוש במראה מקום לחבורה הינו טבעי מאחר ורוב הניווט בבסיס הנתונים מתבצע לפי מופעים.

ג. **מראה מקום לשטח אחסון** (Current of Area) מצביע על הרשומה האחרונה שטופלה על־ידי תוכנית היישום בשטח האחסון של בסיס הנתונים. לכל שטח אחסון יהיה מראה מקום נפרד.

ד. **מראה מקום לתוכנית היישום** (Current of Run-unit) מצביע על המופע של הרשומה האחרונה שטופלה על ידי תוכנית היישום.

עבור תוכנית יישום המשתמשת בתת־סכימה שמכילה את שלשת סוגי הרשומה שבתרשים 23—6 נקבל את מראי המקום הבאים:



פעולות המבוצעות בבסיס הנתונים גורמות לשינוי במראה המקום בהתאם לפקודה המבוצעת. תרשים 24—6 מציג סדרת פעולות המבוצעות בבסיס הנתונים המוצג בתרשים 23—6 ואת תוכן מראי המקום השונים לאחר כל פקודה.

פקודה מבוצעת	RUN-UNIT	DEPARTMENT	TEACHER	SUBJECT	TEACHERS—OF DEPARTMENT	SUBJECTS—OF DEPARTMENT	TEACHES
- קרא רשומת מרצה 104	104	–	104	–	104	–	104
- קרא רשומת אב שלו	מרעי־המחשב	מרעי־המחשב	104	–	מרעי־המחשב	מרעי־המחשב	104
- קרא רשומת נושא שהוא מלמד	ארגון קבצים	מרעי־המחשב	104	ארגון קבצים	מרעי־המחשב	ארגון קבצים	104
- קרא רשומת מרצה הבאה	109	מרעי־המחשב	109	ארגון קבצים	מרעי־המחשב	ארגון קבצים	109

תוכנית — סוג רשומה — חבורה

מראי מקום

תרשים 24 – 6: תוכן מראי מקום בעקבות פעולות בבסיס הנתונים

דוח DBTG מבחין בין **איתור רשומה לבין העברתה לשטח העבודה** (UWA). לשתי פונקציות אלה פקודות נפרדות: פקודת FIND לאיתור רשומה ועדכון מראי המקום המתאימים, ופקודת GET המעבירה לשטח העבודה את הרשומה, שמראה המקום של תוכנית היישום מצביע עליה.

9.2 פקודות FIND לאיתור רשומות

את פקודות האיתור אפשר לחלק לשתי קבוצות: בקבוצה הראשונה תהיינה פקודות המאפשרות איתור ישיר של רשומה בבסיס הנתונים באמצעות פונקציות DIRECT, CALL או אינדקס משני כלשהו. בקבוצה השניה תהיינה פקודות המתייחסות לחבורות ומשתמשות בתוכן מראי המקום. פקודות אלה מאפשרות איתור מופע מסויים של חבורה ו**מיתוג בין חבורות** (Set Switching). המיקום הראשוני בבסיס הנתונים מתבצע באמצעות פקודות איתור מהקבוצה הראשונה ולאחר מכן מתחיל הניווט באמצעות פקודות מהקבוצה השניה.

שתי פקודות FIND הראשונות שייכות לקבוצת פקודות האיתור הראשונה, והשאר שייכות לקבוצת פקודות האיתור השניה.

א. **איתור ראשוני עם מפתח בסיס הנתונים:** איתור ישיר של רשומה בבסיס הנתונים על פי תוכן של מפתח בסיס הנתונים (DB—key).

FIND record-name USING key

במקום המלים record-name, key ירשמו שמות של רשומה ושדה המפתח, בהתאמה.

ב. **איתור ראשוני עם מפתח לוגי:** איתור ישיר של רשומה בבסיס הנתונים על פי תוכן המפתח הלוגי שלה. לפני השימוש בפקודה זו על תוכנית היישום להעביר את תוכן המפתח הלוגי לשדה המפתח של הרשומה.

FIND record-name RECORD

לאיתור רשומת מחלקה שמספרה 135 נשתמש בפקודות הבאות:

העברת המספר 135 לשדה המפתח: MOVE "135" TO DEPARTMENT — ID.

איתור רשומה DEPARTMENT (לפי המפתח): FIND DEPARTMENT RECORD.

ג. **איתור רשומה בהתאם למראי מקום:** הפקודה מאפשרת שימוש בתוכן מראי מקום לאיתור רשומת האב, והיא גם מאפשרת עדכון של מראי מקום. אפשר לכתוב את הפקודה בצורות אחדות בהתאם למראה המקום שבו משתמשים:

```
                                      ⎧ record-name RECORD ⎫
FIND [OWNER IN Set-name OF] CURRENT OF ⎨ set-name    SET    ⎬
                                      ⎨ area-name   AREA   ⎬
                                      ⎩ RUN-UNIT           ⎭
```

המלים set-name, record-name, area-name יוחלפו בשמות רלוונטיים בתוכנית.

הסוגריים המרובעים מציינים אפשרות להשתמש בביטוי כאשר מאתרים רשומת אב. הסוגריים הגדולים מציינים שיש לבחור באחת מהאפשרויות הרשומות בהן. צורת כתיבה זו תחזור בהמשך. לדוגמא, אם התוכנית טיפלה ברשומת "מרצה" ורוצה לאתר את רשומת המחלקה שבה הוא עובד נשתמש בפקודה הבאה:

FIND OWNER IN TEACHERS—OF—DEPARTMFMT OF CURRENT OF RUN—UNIT.

ד. **איתור רשומה המשתתפת בחבורה:** הפקודה מאפשרת ניווט במסגרת החבורה שמשמעו

התקדמות לרשומת הבן הבאה או לרשומת הבן הקודמת, איתור רשומת הבן הראשונה או האחרונה או התקדמות של מספר רשומות בן בחבורה. הפקודה נכתבת בהתאם לאפשרות הניווט שנבחרה:

$$\text{FIND}\left\{\begin{array}{l}\text{NEXT}\\ \text{PRIOR}\\ \text{FIRST}\\ \text{LAST}\\ \text{integer}\end{array}\right\}\ \text{record-name RECORD OF}\ \left\{\begin{array}{l}\text{Set-name SET}\\ \text{Area-name AREA}\end{array}\right\}$$

במקום המלים record-name, set-name, area-name יכתבו שמות רלוונטיים. במקום integer יכתב מספר שלם המציין את מספר הרשומות לקידום.

לדוגמא, אם איתרנו את רשומת המחלקה לפי פקודות שהוצגו בסעיף א׳ לעיל, ניתן לאתר את רשומות המרצה השייכות למחלקה בעזרת הפקודות הבאות:

```
FIND FIRST RECORD OF TEACHERS—OF—DEPARTMENT SET.
— — — — — — — —
FIND NEXT TEACHER RECORD OF TEACHERS—OF—DEPARTMENT SET.
```

ה. **איתור רשומת אב בחבורה:** איתור של רשומת אב של מופע כלשהו בחבורה.

```
FIND OWNER RECORD OF set-name SET
```

אם במהלך סריקת רשומות הנושא דרוש לאתר את רשומת המחלקה שבה הוא נלמד נשתמש בפקודה:

```
FIND OWNER RECORD OF SUBJECTS—OF—DEPARTMENT SET.
```

ו. **איתור רשומה לפי תוכן של שדה.** מיועד לאיתור רשומת בן בחבורה על פי תוכן של אחד או יותר משדות הנתונים המופיעים בה.

```
FIND record-name VIA [CURRENT OF] set-name USING data-item1,.....
```

לאחר איתור רשומת מחלקה עם מפתח 135 אפשר לאתר את רשומת המרצה שתאריך תחילת העבודה שלו הוא x:

```
MOVE "X" TO START—DATE.
FIND TEACHER VIA CURRENT OF TEACHERS—OF—DEPARTMENT
USING START—DATE.
```

אם בסכימה הוגדרה צורת איתור החבורה "מרצים במחלקה" באמצעות המשפט THRU LOCATION MODE OF OWNER, ניתן להטיל את איתור המופע של החבורה על מערכת DBMS. נשתמש בפקודות הבאות:

```
MOVE "135" TO DEPARTMENT—ID.
MOVE "X" TO START—DATE.
FIND TEACHER VIA TEACHERS—OF—DEPARTMENT
        USING START—DATE.
```

ז. **איתור רשומת הבן הבאה על פי תוכן של שדה:** הפקודה מאפשרת סריקה אוטומטית של רשומות הבן עד למציאת רשומה שתוכן אחד או יותר של השדות שלה זהה לערך נתון כלשהו (data-item). מבנה הפקודה:

```
FIND NEXT WITHIN set-name USING data-item1,.....
```

לאחר שמאתרים רשומת מרצה שתאריך תחילת העבודה שלו הוא X, ניתן לאתר את רשומת המרצה הבאה העונה על תנאי זה (START-DATE קבל ערך לאחר פעולת האיתור הראשונה):

```
FIND NEXT WITHIN TEACHERS—OF—DEPARTMENT USING START-DATE.
```

9.3 פקודות נוספות

בנוסף לפקודות הבסיסיות FIND וְ־GET כוללות השפה לטיפול בנתונים את הפקודות הבאות:

— STORE: הוספת רשומה חדשה לבסיס הנתונים. הרשומה מקושרת אוטומטית למופעים של החבורות שבהן יש לה "חברות אוטומטית".

— DELETE: ביטול רשומה קיימת מבסיס הנתונים. לפקודה זו יש מספר אופציות. ניתן להגדיר אופציה שבה תבוטל הרשומה רק אם אין לה רשומות בן בכל החבורות בהן הוגדרה כרשומת אב. קיימת אפשרות לבטל את הרשומה יחד עם כל רשומות הבן שהוגדרו כחברים הכרחיים בלבד, בעוד כל הרשומות שהוגדרו כלא־הכרחיים מנותקים מחברותם בחבורה ונשארים בבסיס הנתונים. האפשרות הראשונה מקבילה לאפשרות הביטול במודל ההיררכי שבה לא ניתן לבטל רשומת אב ללא ביטול קודם של רשומות הבן שלה.

— MODIFY: עדכון תוכן של רשומה קיימת בבסיס הנתונים. יש לאתר תחילה את הרשומה בעזרת אחת מפקודות FIND.

— INSERT: קישור רשומת בן למופע כלשהו של חבורה שבה הרשומה מוגדרת בסוג חברות לא הכרחי. על התוכנית לאתר תחילה את המופע המתאים של החבורה.

— REMOVE: ניתוק רשומת בן ממופע של חבורה שבה היא מוגדרת בסוג חברות לא הכרחי.

10. סיכום

המלצות CODASYL-DBTG מהוות אבן דרך חשובה בהתפתחות מערכות DBMS. המלצות אלה קבעו לראשונה בצורה פורמלית את המושגים סכימה, תת־סכימה, שפה לטיפול בנתונים וכדומה. דוח DBTG מגדיר שלש שפות להגדרת נתונים ושפה אחת לטיפול בהם:

— Schema DDL — שפה להגדרת הסכימה,
— Subschema DDL — שפה להגדרת תת־הסכימה,
— Device/Media Control Language — שפה להגדרת ארגון הנתונים ביחידות האחסון,
— DML — שפה לטיפול בנתונים.

מודל הנתונים הרשתי שמתואר בדוח DBTG מאפשר בניית מבני נתונים מורכבים אשר יכולים לייצג את המציאות במודל הנתונים. שפת DML היא שפה פרוצדורלית המאפשרת את הניווט בתוך בסיס הנתונים. השפה מורכבת ומכילה אוסף גדול של פקודות ומבנים שונים של אותה פקודה.

11. שאלות

1. ציין מהן התכונות של "חבורה" לפי הגדרת CODASYL.
2. לולאות אסורות במודל הרשתי של CODASYL. האם אתה יכול לחשוב על דוגמא שבה איסור זה יכול להוות חיסרון? האם החיסרון משמעותי?
3. מדוע מוגדר המושג AREA? מה המאפיינים שלו? לשם מה הוא מופיע בסכימה?
4. קרא את השאלות הבאות והשב "נכון/לא־נכון":
   — חבורה הינה קשר בעל שם בין סוגי רשומות.
   — סכימה מכילה רק חבורה אחת.
   — עבור כל סוג רשומה הסכימה מגדירה שטח או שטחים שבהם יאוחסנו המופעים של הרשומות.
   — חבורה בסכימה יכולה להיות ללא מיון.
   — ניתן להגדיר סוג רשומה בסכימה כרשומת בן בחבורה אחת או יותר.
   — האפשרות שסוג רשומה יהיה גם "אב" וגם "בן" באותה חבורה אינה נתמכת ע"י DDL.
   — האפשרות שסוג רשומה יהיה גם "אב" וגם "בן" היא שמאפשרת ייצוג רשתות.
   — חבורה סינגולרית מורכבת מרשומת אב ולפחות מרשומת בן אחת.
   — קשר N:M ניתן לייצוג רק באמצעות חבורה אחת.
5. הגדר מהן אפשרויות הגישה והאיתור של רשומה במודל הרשתי. מה היתרונות והחסרונות של כל שיטה?
6. יש לנהל נתונים אודות חברות, מניות ובעלי המניות. הקשרים האפשריים הם:
   — חברה—מניות: רב—רב־ערכי
   — מניות—בעל מניות: רב—רב־ערכי.
   א. הצג את החבורות הדרושות לניהול נתונים אלה.
   ב. כיצד תענה על השאילתות הבאות:
      — מיהם כל בעלי המניות של חברה מסויימת?
      — רשימת המניות שבעל מניות מסויים מחזיק.

פרק ז

מודל הנתונים הטבלאי
(Relational Data Model)

1. מבוא
2. המודל הטבלאי – הגדרה ומושגי יסוד
3. שפות לטיפול ביחסים
4. System R
5. סיכום
6. שאלות

1. מבוא

השלב הקשה ביותר בתכנון בסיס נתונים הינו **קביעת מודל נתונים** המספק הן את דרישות עיבוד הנתונים והן את הצורה שבה רואה הארגון את הנתונים. אם המודל אינו מקיים שני תנאים אלה יוצאים היישום או הארגון נפסדים.

בפרקים הקודמים הוצגו שני מודלים שונים לייצוג נתונים, המודל ההירכי והמודל הרשתי, אשר רוב מערכות DBMS הקיימות היום מבוססות עליהם. קשה לעיתים להציג מודלים אלו, ובפרט המורכבים שבהם, לאנשים שאינם אנשי מקצוע בעיבוד נתונים משום הקשרים המסועפים הקיימים בין הנתונים. למרות פיתוח מערכות עיבוד נתונים המבוססות על מודלים אלו לא פסק המאמץ לחפש דרכים ושיטות להצגה פשוטה יותר של מודל הנתונים עבור המשתמש שאינו מומחה בעיבוד נתונים ולאפשר תכנון פשוט ונוח יותר של המודל. זהו הרקע לפיתוח **מודל הנתונים הטבלאי** (Relational Data Model).

המודל הטבלאי הוצג לראשונה בצורה פורמלית בתחילת שנות ה־70 על ידי E.F. Codd ממעבדות הפיתוח של יבמ. הוא ראה את היתרון העיקרי של המודל באפשרות ההצגה המדוייקת והפשוטה של בסיס הנתונים. הדיוק נובע מכך שהמודל מתבסס על התורה המתמטית של הקבוצות ומשתמש באופרטורים לטיפול **ביחסים** (Relations). הפשטות מושגת על ידי הצגת היחסים במבנה של **טבלה** (Table) המוכר ומובן לכל, ועל ידי טיפול ביחסים באמצעות מספר קטן של אופרטורים פשוטים. אחד המאפיינים החשובים של המודל הטבלאי הוא: **הפעלת אופרטור על טבלה יוצרת טבלה חדשה.** נסביר זאת בהמשך.

בשלב כתיבת הספר מעטות עדיין המערכות המסחריות המשתמשות במודל הטבלאי אך אלו הולכות ורבות. מערכות אלו נמצאות בשלבי פיתוח מואץ ומושקעים מאמצים גדולים על ידי יצרנים שונים להשיג רמת יעילות גבוהה באחסון הנתונים ובגישה אליהם. הקושי העיקרי בהשגת היעילות נובע מכך שהדרישה המרכזית היא להשאיר את תכנון המודל חופשי להכנסת שינויים וגמיש מבחינת אפשרויות הגישה לנתונים. כלומר, למנוע כל מגבלה ולהשיג את מירב היעילות. שתי דרישות אלו **נוגדות** זו את זו וזאת ראינו במודלים הקודמים: גישה יעילה הושגה כאשר "הוקרבה" למענה הגמישות שבאחסנת וניהול הנתונים.

התכונות של המודל הטבלאי הן פשטות תכנון המבנה, גמישות ניהול הנתונים, פשטות וגמישות הגישה לנתונים. תכונות אלו מכוונות את השימוש במודל אל מערכות ויישומים שבהם קשה לקבוע מראש את מבנה הנתונים ואת מסלולי הגישה או היחסים שבין הנתונים. המודל מאפשר גישה אל בסיס הנתונים על־ידי משתמש בלתי מנוסה בעיבוד נתונים באמצעות שפת היחסים הפשוטה והמובנת. הוא יכול לאגור את הנתונים שלו ולאתר אותם לשם הפקת הפלט הדרוש לו בעבודתו.

מערכות המודל הטבלאי אינן מכוונות בשלב זה אל שירותי בסיס הנתונים המרכזי של הארגון הפועל בקצב תנועות גבוה (כדי עשרות ומאות תנועות בדקה אחת). מטרתו אינה עדכון רשומות בקובץ או איתור רשומות בודדות, כפי שמקובל במערכת בנקאית למשל, או במערכת ייצור. המטרה העיקרית היא לתת למשתמש כלי עבודה, אשר יהיה פשוט להבנה ולתפעול, ואשר יאפשר לו לנסח **מהם** הנתונים הדרושים לו ולא **איך** לאחזר אותם מתוך הקובץ. המשתמש במודל הטבלאי **עוסק בהפקת מידע** או דוח, **ואינו מטפל ברשומות** כמקובל במודלים שבהם עסקנו עד עתה.

כיום קיימות מספר מערכות המבוססות על מודל הנתונים הטבלאי אשר מותקנות כבסיס־נתונים מרכזי בארגון לשם שירותים מרכזיים או מחלקתיים. הידועות ביותר מבין מערכות אלו

הן INGRES של חברת Relational Technology, SQL/DS ו־DB2 של חברת IBM, ORACLE של חברת Relational Software, NOMAD של חברת National CSS.

2. המודל הטבלאי – הגדרה ומושגי יסוד

המודל הטבלאי הינו מודל פורמלי לייצוג הישויות והקשרים שביניהן. המודל ההירכי והמודל הרשתי השתמשו בשני מבנים שונים — הרשומה והקשר. **הרשומה** מייצגת את הישות, כלומר את הקשרים שבין התכונות של הישות. **הקשר** מייצג את הקשר בין קבוצות ישות שונות. המודל הטבלאי משתמש במבנה אחד בלבד, **היחס** או **הטבלה**, לייצוג הישויות והקשרים ביניהן ובכך התרומה הגדולה לפשטות המבנה וההבנה.

2.1 יחס (Relation)

נתון אוסף של קבוצות D1, D2,..., Dn לא בהכרח זרות. היחס הינו אוסף מסודר של איברים <d1, d2,...dn>, כך שכל איבר di מתאים לקבוצה Di עבור i=1, 2,...n. ניתן לתאר את היחס כאוסף של **שורות** המחולקות **לעמודות**, או בפשטות — **טבלה** שבה **השורה** מייצגת ישות **והעמודה** מייצגת את התכונות של הישות. סדר העמודות זהה בכל השורות ולכן לכל תכונה יש מקום קבוע בתוך השורה. הערך בכל עמודה נלקח מתוך תחום הערכים האפשריים של אותה עמודה. במודל הנתונים הטבלאי השורה הינה ה"רשומה" והעמודות הן ה"שדות" של הרשומה. כל טבלה הינה יחידה עצמאית ואפשר לראות אותה כ"קובץ" חלקי בתוך בסיס הנתונים.

נסמן את היחס ב־R ונרשום: R(D1,D2,...Dn).

למספר הקבוצות או העמודות ביחס נקרא **המימד של היחס**. לדוגמא, יחס המכיל שתי קבוצות יקרא **יחס בינרי:**

R(D1, D2)

הערכים המופיעים בעמודות נלקחים מתוך תחום הערכים האפשריים של התכונה, ולכן ניתן להגדיר את היחס **כקבוצה חלקית למכפלה הקרטזית של הקבוצות**, או בצורה מתמטית:

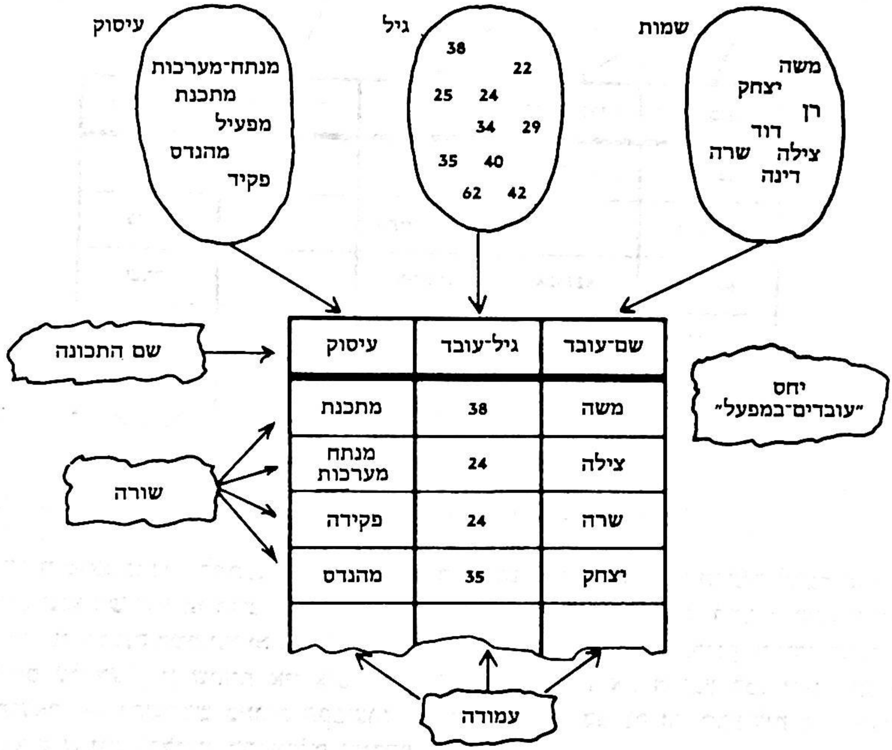
$$R(D1, D2, ...Dn) \subseteq D1 \times D2 \times ...Dn$$

היחס אינו מכיל שתי שורות זהות, כלומר הוא אינו מתיר קיום שתי שורות שונות שמכילות ערכים זהים בכל העמודות שלהן.

נשתמש בדוגמא כדי להסביר את מושג היחס. נגדיר יחס במימד 3 בשם "עובדים־במפעל" המתאר לכל עובד את שלש התכונות הבאות: שם העובד, גיל־עובד ועיסוק. לכל תכונה ניתן להגדיר קבוצה המכילה את אוסף כל הערכים שלה. קבוצת הגיל למשל תכיל את כל המספרים החיוביים בין 18 ל־120 וקבוצת העיסוקים תכיל את כל העיסוקים האפשריים במפעל. אפשר לרשום את היחס בצורה הבאה:

עובדים־במפעל (שם־עובד, גיל־עובד, עיסוק)

נציג את **היחס במבנה טבלה** שבה כל שורה מייצגת עובד אחד וכל עמודה מייצגת תכונה אחת. בכל עמודה מופיע ערך אחד מתוך אוסף הערכים המוגדר בקבוצה של אותה עמודה. כל ערך מופיע בקבוצה פעם אחת בלבד אך הוא עשוי להופיע פעמים אחדות בתוך היחס (הטבלה), בשורות שונות. לכל יחס יש שם המזהה אותו באופן חד־משמעי ולכל עמודה ביחס יש שם המזהה את התכונה המיוצגת בה.

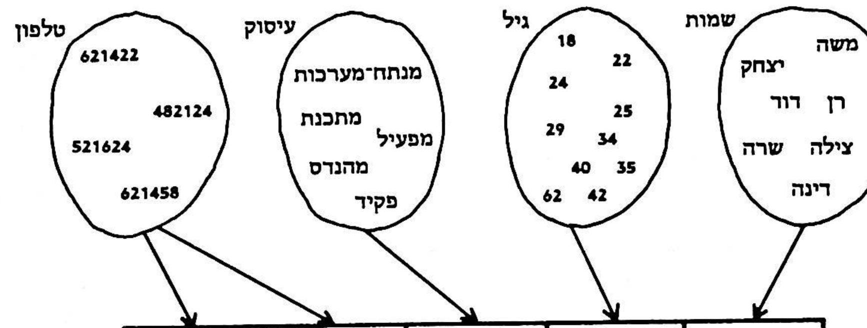


תרשים 1 – 7: יחס "עובדים־במפעל"

עמודות שונות עשויות לקבל את ערכיהן מתוך קבוצת תכונות אחת. לדוגמא, קבוצת מספרי הטלפון מיועדת לעמודה המייצגת את מספר הטלפון בבית העובד וגם לעמודה המייצגת את מספר הטלפון במפעל. לשתי עמודות אלו יש לקבוע שמות שונים המזהים בצורה חד־משמעית את **התפקיד** (Role) של כל אחת מהן. את היחס, הכולל עבור כל עובד את מספרי הטלפון במפעל ובבית, נגדיר כך:

עובדים במפעל (שם־עובד, גיל־עובד, עיסוק, טלפון בבית, טלפון־במפעל)

בתרשים 2—7 מוצג יחס במימד 5. המימד לא נקבע על־ידי מספר הקבוצות של התכונות אלא על־ידי מספר התכונות, או העמודות, המרכיבות את היחס.



שם־עובד	גיל־עובד	עיסוק	טלפון־בבית	טלפון־במפעל
משה	38	מתכנת	482124	621422
צילה	24	מנתח־מערכות	-	621422
שרה	24	פקידה	521624	621458
יצחק	35	מהנדס	-	621422

תרשים 2 – 7: יחס שבו קבוצה אחת משרתת שתי תכונות שונות

היחס מוצג כטבלה למתכנן או למשתמש אך למעשה זהו מבנה פיסי הדומה לקובץ סדרתי פשוט שבו השורות הן הרשומות והעמודות מתאימות לשדות הרשומה. הטבלה הינה מקרה מיוחד של המבנה המתמטי **יחס** שיש לו הגדרה מדוייקת יותר מהגדרת הקובץ. המודל הטבלאי מבוסס על העיקרון שתחת אילוצים מסויימים ניתן לראות את הקובץ כמו יחס מתמטי וכתוצאה — להשתמש בתורת הקבוצות והיחסים לטיפול במבנה זה. הפעולות המבוצעות בקבצים שונות לחלוטין מהפעולות המבוצעות ביחסים.

הייחוד של המודל הטבלאי נובע מהשימוש ביחס לייצוג הישות ולייצוג הקשר בין קבוצות ישות שונות. כל שורה ביחס מייצגת את הקשרים בין התכונות של ישות אחת. **הקשר** בין שתי קבוצות ישות שונות יתבצע על־ידי קיום שתי תכונות זהות בשני היחסים. נתבונן לדוגמא ביחס "מחלקות" וביחס "עובדים־במפעל" אשר מתוארים בשתי טבלאות נפרדות בתרשים 3—7 ומנוסחים כך:

מחלקות (מספר־מחלקה, שם־מחלקה, תקציב)

עובדים־במפעל (שם־עובד, גיל־עובד, עיסוק)

מכיון שכל עובד שייך למחלקה מסויימת ניתן להוסיף ליחס "עובדים־במפעל" את התכונה "מספר מחלקה" אשר תייצג את המחלקה שבה מועסק העובד. עתה אפשר ליצור את הקשר בין היחסים, או הטבלאות, באמצעות התכונה "מספר מחלקה" שמופיע בכל אחד מהם. בטבלאות שלפנינו אין כל מיון באיזו שהיא עמודה מכיון שהמודל אינו מחייב לעשות זאת. בהמשך נבחן את כדאיות המיון לפי מפתח לשם יעול אחזור הנתונים.

מחלקות

מספר־מחלקה	שם־מחלקה	תקציב
10	אבטחת איכות	100.000
12	פיתוחי תוכנה	2.000.000
17	ייצור	1.500.000

עובדים־במפעל

שם־עובד	גיל־עובד	עיסוק	מספר־מחלקה
משה	38	מתכנת	12
צילה	24	מנתח־מערכות	12
שרה	24	פקידה	10
יצחק	35	מהנדס	12

תכונה משותפת

תרשים 3 – 7: קשר בין קבוצות ישות במודל הטבלאי

2.2 מפתחות (Keys)

בכל יחס יש להגדיר תכונה אחת, או יותר, אשר מזהה את השורה בצורה חד־משמעית. לתכונה או מספר תכונות אלה נקרא **מפתח**. המפתח מקבל ערכים שונים בכל שורה ולכן אין שתי שורות המכילות את אותו הערך של המפתח והזיהוי שלהן הוא חד־משמעי. ניתן לומר שהמיפוי בין המפתח לבין תחום הערכים של המפתח הוא מיפוי חד—חד־ערכי. אם המפתח מורכב ממספר תכונות נאמר שזהו **מפתח מורכב** (Compound Key).

אפשרי שמספר תכונות תזהיינה בצורה חד־משמעית את הישות, כלומר קיימים מספר מפתחות שונים אשר נגדירם בשם **מפתחות אפשריים** (Candidate Keys). מבין כל המפתחות האפשריים נבחר בצורה שרירותית מפתח אחד אשר יהיה **מפתח ראשי** (Primary Key) ויזהה את הישות או את השורה בטבלה. מפתח ראשי חייב לקבל ערך ממשי ואינו יכול לקבל **ערך בלתי מוגדר** (Null Value), אשר אינו מופיע בקבוצה של התכונה. שאר התכונות יכולות לקבל ערכים "בלתי מוגדרים". מקובל לסמן קו מתחת התכונה המשתמשת המפתח ראשי.

נגדיר יחס "סטודנטים" שבו התכונות הבאות:

סטודנטים (מספר זהות, מספר־סטודנט, שם־סטודנט, כתובת)

אפשר לבחור ביחס זה במספר מפתחות אפשריים כגון מספר זהות, מספר סטודנט, שם סטודנט וכתובת (מתוך הנחה שאין שני סטודנטים בעלי שם זהה הגרים באותה כתובת). מבין כל המפתחות האפשריים האלה אפשר לבחור במפתח ראשי "מספר סטודנט" אשר אינו יכול לקבל ערכים בלתי מוגדרים ואינו מורכב:

סטודנטים (מספר־זהות, <u>מספר־סטודנט</u>, שם־סטודנט, כתובת)

סטודנטים

מספר־זהות	מספר־סטודנט	שם־סטודנט	כתובת
84245678	12456	איל גפן	השרון 4, תל־אביב
61245682	34254	שרה ראובני	אוליפנט 8, תל־אביב
32451276	61245	אבי שגיב	שדה 12, חיפה
44524125	72242	רון כהן	הרצל 48, חיפה·

מפתח ראשי

מפתחות אפשריים

תרשים 4 – 7: מפתחות אפשריים ומפתח ראשי ביחס "סטודנטים"

לעיתים אין ביחס תכונה כלשהי אשר יכולה לזהות את הישות באופן חד־משמעי. במקרה זה אי אפשר לקבוע מפתח ראשי מתוך התכונות הקיימות ויש להוסיף תכונה נוספת, **יזומה**. נדגים זאת באמצעות היחס "הזמנת־לקוח" המייצגת את ההזמנות שמבצעים הלקוחות של המפעל:

הזמנת־לקוח (מספר־לקוח, שם־לקוח, תאריך־הזמנה, ערך־הזמנה)

אף לא אחת מתכונות אלו מזהה בצורה חד־משמעית את הישות. ללקוח אחד עשויות להיות הזמנות רבות והוא יכול לבצע מספר הזמנות באותו תאריך ובאותו סכום. נוסיף את התכונה היזומה "מספר־הזמנה" ונגדיר אותה כמפתח ראשי. היחס יוצג עתה בצורה הבאה:

הזמנת־לקוח (<u>מספר־הזמנה</u>, מספר־לקוח, שם־לקוח, תאריך־הזמנה, ערך־הזמנה)

הזמנת־לקוח

מספר־הזמנה	מספר־לקוח	שם־לקוח	תאריך־הזמנה	ערך־הזמנה
159	124	תדיראן	12.2.82	1,000
320	324	טלרד	13.1.82	250,000
425	124	תדיראן	12.2.82	300,000
160	124	תדיראן	13.8.81	1450
617	350	תדיראן	15.2.92	500,000

מפתח ראשי (יזום)

תרשים 5 – 7: מפתח ראשי כתכונה יזומה

תכונה המשתתפת במפתח אפשרי אחד לפחות תקרא **תכונה עיקרית** (Prime Attribute) ותכונה שאינה משתתפת באף מפתח אפשרי תקרא **תכונה לא עיקרית** (Non-Prime Attribute). לדוגמא, ביחס "הזמנת־לקוח" רק התכונה "מספר הזמנה" תקרא תכונה עיקרית וכל שאר התכונות תקראנה תכונות לא־עיקריות.

לסיכום נגדיר בצורה פורמלית יותר את המפתח. מפתח K של יחס R הינו תכונה אחת או יותר המקיימות את התנאים הבאים:

— **זיהוי חד משמעי** (Unique Identification): בכל שורה של היחס R הערך של K מזהה בצורה חד־משמעית את השורה,

— **אי כפילות** (Nonredundancy): אם המפתח K הוא מפתח מורכב לא ניתן להוציא מתוך המפתח תכונה המשתתפת בו מבלי לפגוע בזיהוי החד־משמעי.

3. שפות לטיפול ביחסים

מודל הנתונים הטבלאי מארגן את כל הנתונים בצורת **טבלאות** (יחסים) ומגדיר **שפת נתונים לטיפול ביחסים** (Relational Data Language). **האופרטורים** של השפה מאפשרים בניית יחסים חדשים מתוך יחסים קיימים. ליחסים החדשים זכות קיום משל עצמם ברגע שהם נוצרו וניתן לטפל בהם באמצעות אותם אופרטורים, כמו שמטפלים ביחסים הרגילים.

השפות לטיפול ביחסים שייכות לאחת משתי קבוצות:

— **שפות אלגבריות** מגדירות אוסף של אופרטורים שניתן להפעיל על היחסים לקבלת יחסים חדשים. המשתמש מפעיל את האופרטורים השונים על היחסים בסדר לוגי מסויים עד לקבלת יחס שמכיל את הנתונים הדרושים.

— **שפות עיליות** מאפשרות למשתמש להגדיר כל שהוא רוצה ומשאירות את ההחלטה על שיטת הביצוע בידי מערכת DBMS. מקובל גם לקרוא לשפות אלה בשם הכולל — Relational Calculus. היתרון שלהן הוא בכך שהמשתמש אינו מגדיר כיצד להגיע לנתונים הדרושים והוא מאפשר למערכת לבצע אופטימיזציה לקבלת אחזור יעיל. החסרון של שפות אלה הוא הקושי ביישום שלהן.

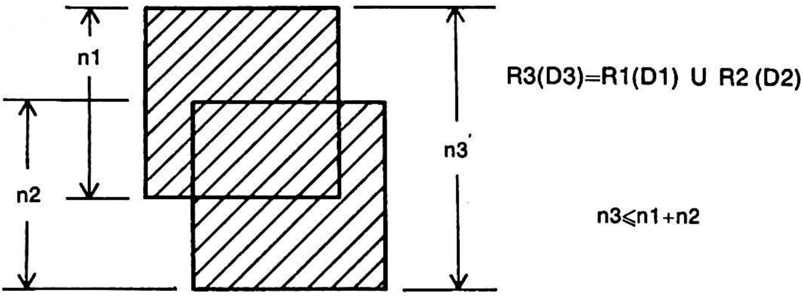
מערכות SQL/DS, QBE ואחרות מהוות יישום של גישות אלו. במערכות מסחריות נמצא בדרך כלל שילוב של שפה אלגברית ושפה עילית.

3.1 אלגברה של יחסים (Relational Algebra)

שפת הנתונים האלגברית לטיפול ביחסים מגדירה אוסף של אופרטורים המאפשרים את הטיפול ביחסים קיימים לשם קבלת אוסף של יחסים חדשים. אופרטורים אלה יכולים להיות **איחוד** (או חיבור), **חיסור**, **היטל** וצירוף. האופרטורים פועלים על יחס אחד או על שני יחסים בהתאם לענין.

3.1.1 איחוד יחסים (Union)

איחוד שני יחסים מגדיל את התחום של תכונה זהה בשני היחסים ומייצר יחס שלישי. שורות זהות בשני היחסים יוצרות רק שורה אחת ביחס החדש. נסמן את אופרטור האיחוד ב־U. האיחוד יכול להתבצע בין שני יחסים בלבד ולכן איחוד של יותר משני יחסים נעשה בשלבים: איחוד של שני יחסים ואיחוד התוצאה עם היחס השלישי.



תרשים 6 - 7: איחוד שני יחסים

נבחן דוגמא לאיחוד שני יחסים. R1 מגדיר את כל המוצרים המיועדים לשוק המקומי ו־R2 מגדיר את כל המוצרים המיועדים ליצוא.

R1 = **שוק־מקומי** (קוד־מוצר, שם־מוצר)

R2 = **יצוא** (קוד־מוצר, שם־מוצר)

חלק מהמוצרים המיועדים לשוק המקומי מיועדים גם ליצוא ולכן קיימות שורות זהות בשני היחסים. נגדיר יחס חדש R3 שהינו **איחוד** של שני היחסים R1 ו־R2. R3 מייצג את כל המוצרים שהארגון מייצר ואשר מיועדים לשוק המקומי ולייצוא. מוצרים המופיעים בשני היחסים יופיעו פעם אחת בלבד ביחס החדש.

R1

קוד־מוצר	שם־מוצר
1245	מוצר־1
1832	מוצר־3
2421	מוצר־8
2534	מוצר־9

R2

קוד־מוצר	שם־מוצר
1832	מוצר־3
1995	מוצר־4
1982	מוצר־6
2534	מוצר־9

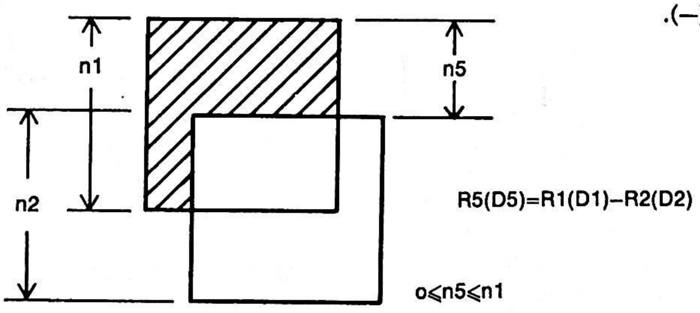
R3=R1U R2

קוד־מוצר	שם־מוצר
1245	מוצר־1
1832	מוצר־3
1995	מוצר־4
1982	מוצר־6
2421	מוצר־8
2534	מוצר־9

תרשים 7 – 7: איחוד שני יחסים

3.1.2 חיסור יחסים (Difference)

בפעולת חיסור בין שני יחסים מורידים מיחס אחד את השורות הזהות המופיעות גם ביחס השני, ומעבירים ליחס החדש את השורות הנותרות בלבד (אלו הן השורות מהיחס הראשון). סדר היחסים חשוב בפעולה זו כמקובל בכל פעולת חיסור אלגברית. את אופרטור החיסור נסמן ב(—).



תרשים 8 – 7: חיסור יחסים

דוגמא לחיסור שני יחסים ניתנת בתרשים 9—7. היחס R5 הינו ההפרש בין היחס R1 ליחס R2 ומייצג את כל המוצרים המיועדים לשוק המקומי בלבד. מוצרים המיועדים לשוק המקומי ולשוק היצוא לא יופיעו ביחס החדש (מוצר־3 ומוצר־9).

R1

קוד־מוצר	שם־מוצר
1245	מוצר־1
1832	מוצר־3
2421	מוצר־8
2534	מוצר־9

R2

קוד־מוצר	שם־מוצר
1832	מוצר־3
1995	מוצר־4
1982	מוצר־6
2534	מוצר־9

R5=R1−R2

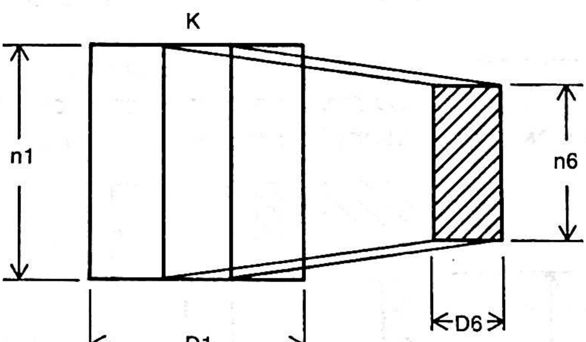
קוד־מוצר	שם־מוצר
1245	מוצר־1
2421	מוצר־8

תרשים 9 - 7: דוגמא לחיסור יחסים

3.1.3 היטל (Projection)

פעולת ההיטל מאפשרת להוציא עמודות (או תכונות) מתוך יחס קיים ולבנות מהן יחס חדש. את העמודות שהוצאו אפשר לסדר בכל סדר רצוי. פעולת ההיטל מבטלת שורות זהות, אם נוצרות כאלו, ביחס החדש. את אופרטור ההיטל מסמנים ב־Π.

$$R6(D6) = \Pi(K \text{ of } R1(D1)]$$



תרשים 10 - 7: היטל של יחס

לדוגמא נניח שקיים יחס R1 המכיל שלש תכונות:

R1 = **עובד** (שם־עובד, גיל־עובד, עיסוק)

פעולות ההיטל מאפשרת שליפת תכונה אחת או יותר ובניית יחס חדש שמכיל רק תכונות אלה. נגדיר יחס R2 המהווה היטל של R1 ומכיל רק את שם העובד ואת עיסוקו. בצורה דומה ניתן להגדיר יחס R3 שהוא היטל של R1 ומכיל רק את העיסוקים במפעל. ביחס R3 מופיע כל עיסוק פעם אחת בלבד מכיוון שאופרטור ההיטל מבטל את השורות הזהות.

R1

שם־עובד	גיל־עובד	עיסוק
משה	24	מתכנת
צילה	24	מנתח־מערכות
שרה	22	פקידה
יצחק	34	מהנדס
דוד	36	מתכנת
רון	32	מהנדס

$R2 = \pi R1$ (שם־עובד, עיסוק)

שם־עובד	עיסוק
משה	מתכנת
צילה	מנתח־מערכות
שרה	פקידה
יצחק	מהנדס
דוד	מתכנת
רון	מהנדס

$R3 = \pi R1$(עיסוק)

עיסוק
מתכנת
מנתח־מערכות
פקידה
מהנדס

תרשים 11 - 7: דוגמא להיטל יחסים

3.1.4 צירוף יחסים (Join)

פעולת הצירוף בונה יחס חדש מתוך שנים או יותר יחסים נתונים על פי **תכונה אחת** משותפת בשניהם. אל היחס החדש מועברות שורות שבהן נמצאה התכונה המשותפת בפעולה הנקראת **צירוף טבעי** (Natural Join) והן מכילות את צירוף התכונות משני היחסים. ניתן להגדיר פעולות צירוף שמעבירות אל היחס החדש שורות המכילות ערכים לא שווים, ערכים גדולים או ערכים קטנים מערך כלשהו באחד היחסים. נסמן את אופרטור הצירוף ב־★.

D0 — תכונה משותפת   R7(D7) = R1 ★ R2 (D0)

דוגמא לצירוף טבעי של שני יחסים ניתנת בתרשים 12—7. היחס החדש R3 הוא צירוף של שני היחסים R1 ו־R2 אשר נעשה על פי התכונה המשותפת ״מספר מחלקה״. ביחס החדש מופיעות רק השורות שבהן הערכים של מספר המחלקה שווים בשני היחסים. מחלקה שמספרה 139 למשל אינה מופיעה ביחס החדש כי אינה מופיעה ביחס R1. ביחס החדש מופיעות כל התכונות משני היחסים ש״צורפו״.

R1 —עובד

שם־עובד	מספר־מחלקה	עיסוק
משה	124	מתכנת
צילה	124	מנתח־מערכות
שרה	135	פקידה
יצחק	137	מהנדס

R2 — מחלקות

מספר־מחלקה	שם־מחלקה
124	פיתוח תוכנה
135	מזכירות
137	הנדסת ייצור
139	קלט/פלט

R3 = R1 * R2 (מספר־מחלקה)

שם־עובד	מספר־מחלקה	שם־מחלקה	עיסוק
משה	124	פיתוח תוכנה	מתכנת
צילה	124	פיתוח תוכנה	מנתח־מערכות
שרה	135	מזכירות	פקידה
יצחק	137	הנדסת יצור	מהנדס

תרשים 12 - 7: דוגמא לצירוף של יחסים

ניתן להגדיר פעולת הצירוף שמעבירה ליחס החדש את כל השורות משני היחסים. אם לערך מסויים ביחס אחד אין מקביל ביחס האחר ישארו חלק מהתכונות ביחס החדש ריקות, מקובל לקרוא לצירוף זה צירוף מלא (Full Join). היחס החדש יכיל את כל התכונות משני היחסים המקוריים.

פעולת הצירוף הטבעי מאפשרת שליפת שורות מסויימות מתוך יחס מקורי. נניח שיש לקבל את רשימת כל העובדים במחלקת ״פיתוח תוכנה״. לשם כך נבנה יחס עזר R4 אשר מכיל שורה אחת בלבד ונצרף אותו אל יחס R1. התוצאה היא יחס R5 אשר מכיל רק את העובדים במחלקת פיתוח־תוכנה. גם כאן מכיל היחס החדש את התכונות משני היחסים המקוריים.

שם-עובד	מספר-מחלקה	שם-מחלקה	עיסוק
משה	124	פיתוח תוכנה	מתכנת
צילה	124	פיתוח תוכנה	מנתח-מערכות
שרה	135	מזכירות	פקידה
יצחק	137	הנדסת ייצור	מהנדס
-	139	קלט/פלט	-

תרשים 13 – 7: צירוף מלא של יחסים מתרשים 12 – 7

R1 עובד

שם-עובד	מספר-מחלקה	עיסוק
משה	124	מתכנת
צילה	124	מנתח-מערכות
שרה	135	פקידה
יצחק	137	מהנדס

R4 יחס-עזר

מספר-מחלקה	שם-מחלקה
124	פיתוח תוכנה

R5=R1★R4 (מספר-מחלקה)

שם-עובד	מספר-מחלקה	שם-מחלקה	עיסוק
משה	124	פיתוח תוכנה	מתכנת
צילה	124	פיתוח תוכנה	מנתח-מערכות

תרשים 14 – 7: צירוף טבעי עם יחס עזר

אם רוצים לקבל רק את שמות העובדים והעיסוקים שלהם במחלקת פיתוח תוכנה ניתן לבצע היטל על היחס החדש ולהוציא מתוכו רק את העמודות הדרושות. התוצאה של ביצוע פעולה כלשהי ביחסים מפיקה יחס חדש שעליו ניתן לפעול כמו על היחסים המקוריים שמהם הוא נבנה, ולפי תכונה זו אפשר לענות על דרישה זו. אפשר להשיג את התוצאה המבוקשת על ידי שילוב האופרטורים של צירוף והיטל בהוראה אחת:

R6=π(R1 ★ R4 (מספר-מחלקה)) [שם-עובד, עיסוק]

3.1.5 דוגמאות נוספות לפעולות ביחסים

דוגמא לשמוש באופרטורים צירוף והיטל לשליפת נתונים מתוך מודל הנתונים הטבלאי ניתנת בתרשים 15—7 שבו מוצגים שלשה יחסים. יחס R1 מגדיר את כל הפריטים שהמפעל רוכש, יחס R2 מגדיר את כל הספקים לפריטים אלו. יחס R3 מגדיר את הקשר הרב—רב־ערכי בין פריט לספק ומציין את הכמות של כל פריט המסופקת על ידי כל ספק.

R1 פריטים

מספר־פריט	שם־פריט	מחיר
P	PN	PR
P1	מברג	50
P2	בורג 2	1
P3	ציר עגול	400
P4	ציר מרובע	420
P5	בורג 4	3
P6	ברזל 5	120

R2 ספקים

מספר־ספק	שם־ספק	עיר
S	SN	C
S1	משה	תל־אביב
S2	יצחק	ירושלים
S3	דוד	ירושלים
S4	רון	חיפה
S5	שרון	תל־אביב

R3 פריט־ספק

מספר־פריט	מספר־ספק	כמות
P	S	QT
P1	S_1	100
P2	S_1	120
P3	S_1	40
P4	S_2	50
P5	S_2	100
P4	S_3	70
P5	S_3	90
P6	S_1	100
P6	S_2	90

תרשים 15 – 7: בסיס נתונים טבלאי במפעל

דרישות העיבוד:

— שמות הספקים המספקים פריטים למפעל: נבצע היטל על היחס R2 ונשלוף רק את העמודה של שם הספק.

R2[SN]

— שמות הספקים הפועלים בתל־אביב: נבנה יחס עזר R4 שמכיל רק את שם העיר המבוקשת ונבצע צירוף עם היחס R2.

R4 יחס עזר

עיר
C
תל־אביב

$$\Pi\,(R2 \bigstar R4(C))\,[SN]$$

— אילו פריטים ניתן להשיג בכל עיר ועיר: נבצע תחילה צירוף של יחס R2 ליחס R2 לקבלת רשימת הפריטים המסופקים על ידי כל ספק. על היחס החדש הנה נבצע היטל לשליפת העמודות של מספר פריט ועיר כמבוקש.

$$\Pi\,(R2 \bigstar R2\,(S\neq))\,[P\neq, C]$$

פעולת ההיטל מצמצמת שורות זהות ולכן ביחס הסופי יופיע כל פריט פעם אחת בלבד עבור כל עיר שבה ניתן לקנותו.

— מהם מחירי הפריטים שמספק ספק S3: נבנה יחס עזר R4 שמכיל רק את ספק S3. נבצע חיתוך בינו לבין היחס R3 ונקבל את רשימת כל הפריטים המסופקים על ידי ספק S3. עתה נבצע צירוף של היחס החדש עם היחס R1 לקבלת הנתונים הנמצאים בטבלה זו. לבסוף נבצע היטל לקבלת עמודות הפריט והמחיר בלבד.

$$\Pi(\Pi(R4 \star R3\ (S\neq))\ [P\neq])\star R1\ (P\neq))\ [P\neq, PR]$$

לחזרה, האם תוכל לבנות את טבלאות הנתונים עבור כל היחסים שהוצגו בסעיף זה?

3.2 חשבון של יחסים (Relational Calculus)

בסעיף קודם הוצגה האלגברה של היחסים המגדירה אוסף של אופרטורים אשר המשתמש מפעיל על יחסים נתונים לשם קבלת היחס המבוקש. סדר הפעולות לקבלת היחס המבוקש, התוצאה, נקבע על ידי המשתמש.

בחשבון יחסים המשתמש מגדיר את היחס המבוקש ואינו מציין את הדרך שבה יש לפעול עד לקבלתו. שיטה זו מאפשרת לו לציין מה הוא רוצה לקבל ולא איך הוא רוצה לקבל את התוצאה. על מערכת DBMS לקבוע את השלבים הדרושים ואת האופרטורים של אלגברת היחסים שיש להפעיל לקבלת היחס המבוקש.

למשתמש בשפת חשבון יחסים יש **שטח עבודה** (Workspace) אחד או יותר המשמשים לתקשורת עם מערכת DBMS. המערכת שולפת את השורות המתאימות מתוך היחסים ומעבירה אותם לשטח העבודה לטיפול המשתמש בשפת השאילתא או באמצעות השפה המארחת. לכל משתמש במערכת שטח עבודה משלו, כמקובל בשיטות עבודה במערכת של **עיבוד רב־משימות** (Multitasking), ולכן הטיפול ביחסים הנמצאים בשטח העבודה אינו יוצר התנגשות עם משתמשים אחרים.

E.F.Codd הציג בשנת 1972 שפה בשם ALPHA המבוססות על חשבון יחסים והראה שניתן להפוך כל שאילתא המוצגת בצורת חשבון יחסים לביטוי מתאים באלגברה של יחסים. מערכת INGRES שפותחה באוניברסיטת ברקלי בקליפורניה מבוססת על שיטה זו.

3.2.1 הצגת שאילתא בחשבון של יחסים

שאילתא בחשבון יחסים מורכבת משני חלקים: חלק המציג את התכונות (עמודות) של היחס שעל המערכת לשלוף וחלק המציג את התנאי הלוגי שעל השורות לקיים על מנת שתבחרנה מתוך היחס.

נציג תחילה מספר סימונים מקובלים בחשבון יחסים:

גם	AND	$\wedge$
או	OR	$\vee$
לא	NOT	$\neg$
שווה	equal	$=$
לא שווה	not equal	#
קטן מ־	less than	$<$
קטן מ־או שווה ל־	less or equal	$\leqslant$

>	greater than	גדול מ־
⩾	greater or equal	גדול מ־או שוה ל־
:	such that —	כך ש־ (*)
∃	Existential quantifier	קיים
∀	Universal quantifier	עבור כל...

(*) הביטוי בצד שמאל של הנקודותיים מציין מה יש לשלוף מבסיס הנתונים והביטוי מצד ימין מציין את תנאי השליפה

כדי לציין שם של עמודה בשאילתא יש לציין תחילה לאיזה יחס היא שייכת: "מספר־ספק" ביחס ספקים ירשם SUP. SUP-NO. אם לא משתמשים בשמות זהים לעמודות ביחסים שונים אין צורך להשתמש במציין שם־היחס (Qualifier). אין צורך לרשום מציין לעמודה PRICE מכיון שהשם PRICE נמצא ביחס אחד בלבד, וכן ITEM ביחס ORD, אשר אינו מופיע בכל מקום אחר. השאילתות שבדוגמאות מנוסחות בצורה מתמטית פורמלית. במערכות מסחריות פותחו שפות שאילתות פשוטות יותר כפי שניווכח בהמשך.

ITEM

מספר־פריט	שם־פריט	מחיר
ITEM-NO	ITEM-NAME	PRICE
P1	מברז	50
P2	בורג 2	1
P3	ציר עגול	400
P4	ציר מרובע	220
P5	בורג 4	3
P6	ברזל 5	80

SUP

מספר ספק	שם־ספק	עיר
SUP-NO	SUP-NAME	CITY
S1	משה	תל־אביב
S2	יצחק	ירושלים
S3	דוד	ירושלים
S4	רנן	חיפה
S5	שרון	תל־אביב

ORD

מספר־פריט	מספר־ספק	כמות
ITEM	SUP-NO	QNTY
P1	S1	100
P2	S1	120
P3	S1	40
P4	S2	50
P5	S2	100
P4	S3	70
P5	S3	90
P6	S1	100
P6	S2	90

תרשים 16 - 7: בסיס נתונים טבלאי לשאילתות

נשתמש בטבלאות המוצגות בתרשים 16—7 להדגמת שאילתות המבוססות על חשבון יחסים.

א. דרושים "מספרי ספק" של כל הספקים של הפריט P1.

את התוצאה נקבל ביחס R. כלומר היחס R יכיל את העמודה "מספר ספק" עבור כל הפריטים שמספרם P1.

R (ORD.SUP-NO): ORD.ITEM-NO = 'P1'

ב. דרושה רשימת שמות הספקים המספקים את פריט P1.

R (SUP. SUP-NAME):∃ ORD (ORD. SUP-NO =
SUP. SUP-NO ∧ ORD. ITEM = 'P1')

השאילתא מבקשת לשלוף את כל שמות הספקים (SUP-NAME) מתוך היחס SUP שעבורם קיימת שורה ביחס ORD שבו הערך של מספר הספק שווה למספר הספק ביחס SUP והערך של מספר פריט שווה לערך P1. דוגמא זו מציגה קשר בין שני היחסים ספק והזמנה.

ג. דרושה רשימת שמות הספקים שאינם מספקים את פריטים P1 ו־P3.

R (SUP.SUP-NAME) :¬ ∃ ORD (ORD.SUP-NO =
SUP.SUP-NO ∧ (ORD.ITEM= 'P1' V
ORD.ITEM='P3'))

ד. דרושה רשימת שמות הספקים והערים, עבור כל הספקים של פריטים שמחירם נמוך מ־100.

R(SUP. SUP-NAME, SUP.CITY): ∃ ITEM (ITEM.ITEM-NO =
ORD. ITEM ∧ ORD.SUP-NO=SUP.ORD-NO ∧
ITEM.PRICE<100)

3.2.2 שפת ALPHA

שיטת רישום מתמטית של השאילתא אינה נוחה למשתמש ולכן פותחה, כפי שכבר צויין, שפת ALPHA המבוססת על חשבון יחסים אבל נוחה יותר מנקודת מבט המשתמש. נציג כאן בקצרה את שפת ALPHA, שהיא אחת מהשפות לטיפול בבסיס נתונים טבלאי. מבנה כללי של פקודת השליפה:

GET INTO workspace-name (target-list) [WHERE option-list]

workspace-name: שם שטח העבודה שאליו יש להעביר את התשובה,
target-list: אוסף של שמות עמודות אשר את הערכים שלהן יש להעביר לשטח העבודה. לפני שם עמודה יש לציין את שם היחס שבו העמודה מופיעה,
option-list: רשימת תנאי השליפה ואופציות של מיון היחס שנשלף לתוך שטח העבודה. התנאי לא ירשם אם אין בו צורך, כמו בדוגמא הראשונה שלהלן.

דוגמאות לשאילתות בשפת ALPHA:

א. דרושים שמות הספקים במערכת.

GET INTO TBL (SUP.SUP-NAME)

ב. דרושים מספרי פריט ושם־פריט של פריטים שמחירם נמוך מ־100.

GET INTO TBL (ITEM. ITEM-NO, ITEM. ITEM-NAME)
WHERE (ITEM. PRICE < 100)

ג. דרושים מספרי הספק של כל הספקים של פריט P1.

GET INTO TBL (ORD. SUP—NO)

```
WHERE (ORD. ITEM = 'P1')
```

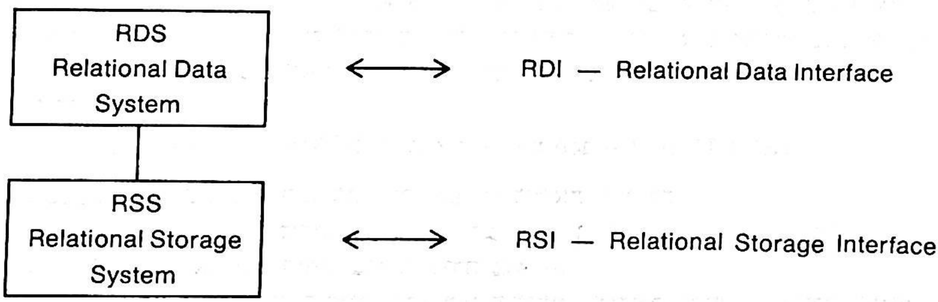
ד. דרושה רשימת שמות הספקים של הפריט P1.

```
GET INTO TBL (SUP.SUP-NAME)
        WHERE ∃ ORD (ORD.SUP-NO = SUP.SUP-NO  AND
                ORD.ITEM = 'P1'
```

SYSTEM R .4

SYSTEM-R היא מערכת נסיונית של חברת יבמ אשר פותחה במעבדות המחקר שלה ב־San Jose ככלי מחקר של מערכות ניהול בסיסי נתונים טבלאיים. שפת SEQUEL של מערכת זו הוכרזה על ידי יבמ בראשית 1981 עבור המערכת לניהול בסיסי נתונים — SQL/DS (Structured Query Language/Data System), וב־1983 — עבור המערכת DB2 (Data Base 2). בהמשך נציג את הנושאים המרכזיים בארכיטקטורה של מערכת מעניינת זו.

4.1 מרכיבי המערכת



תרשים 17 - 7: מבנה פונקציונלי של SYSTEM-R

SYSTEM-R מציגה בפני המשתמש מבנה טבלאי. בתוכה היא מורכבת משתי תת־מערכות RDS ו־RSS **ושני מישקים** (Interfaces) עיקריים RDI ו־RSI.

תת־המערכת RSS מטפלת בניהול ובאחסון הטבלאות ביחידות האחסון, בהקצאת מקום אחסון, הקצאת מאגרי קלט ופלט וגיבוי התנועות ובוחרת את מסלולי הגישה היעילים אל הנתונים שבטבלאות.

כדי להשיג יעילות בביצוע הגישות לנתונים יש להשתמש בטכניקות שתאפשרנה גישה ישירה. אלו מושגות באמצעות מצביעים אשר יוצרים למעשה מבנה של רשתות, ובאמצעות אינדקסים אשר נקבעים על־ידי מתכנן בסיס הנתונים. בכך מוצג המודל הפנימי של המערכת כמודל רשתי על אף המופע החיצוני שלו כמודל טבלאי. לסיכום נאמר שתת־המערכת RSS

הינה שיטת גישה משוכללת אשר מנהלת את הטבלאות המאוחסנות בבסיס הנתונים.

הגישה אל תת־המערכת RSS הינה באמצעות המישק RSI אשר מתווך בין המערכת לבין העולם החיצון. הוא מורכב מאוסף של אופרטורים שניתן להפעילם ישירות מתוך שפות מארחות כגון COBOL או PL/1.

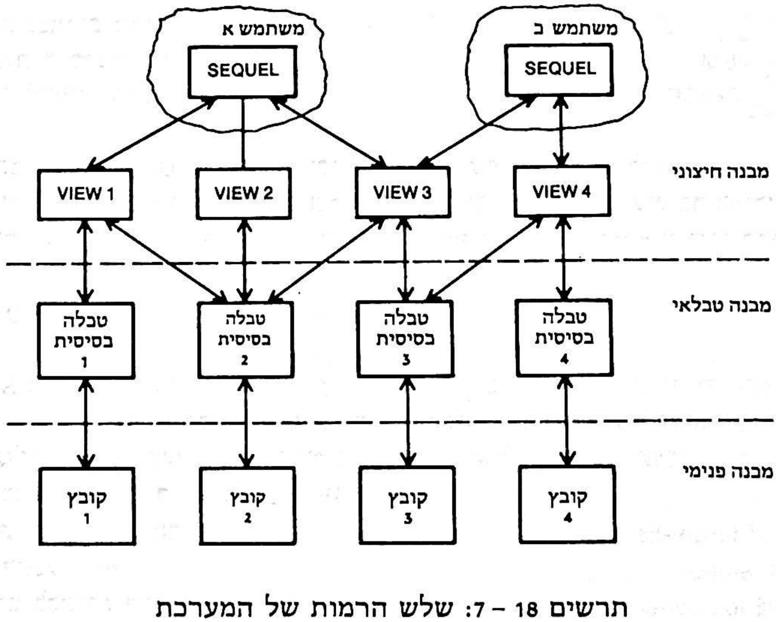
השפת העילית SEQUEL (Structured English Query Language) היא חלק מ־RDI ומאפשרת את הטיפול בטבלאות. הפקודות שלה מתורגמות לשפת מכונה באמצעות קדם־מהדר של RDS אשר גם מבצע אופטימיזציה של מסלולי הגישה לנתונים.

כל פקודות המכונה המטפלות בגישה לנתונים עבור תוכנית מסויימת נקראות **מודול הגישה** (Access Module) ומאוחסנות בבסיס הנתונים. בזמן ריצת התוכנית נשלף מודול הגישה מבסיס הנתונים לשם ביצוע פעולות קלט ופלט.

4.2 הגדרת נתונים

מבנה הנתונים העיקרי במערכת הוא **הטבלה הבסיסית** (Base Table). הגדרת הטבלאות והכנסת שינויים בהן נעשות בצורה פשוטה, נוחה וברורה למשתמש. במונחים של CODASYL הטבלה הבסיסית מקבילה לסוג הרשומה.

המשתמש יכול לפעול על הטבלאות הבסיסיות, אך במקרים רבים נוח לו לפעול על **מבנה מופשט** אשר מגדיר את זוית הראייה או **המבט** שלו (View) על הנתונים. הוא יעדיף את הדרך הזו כאשר ברצונו לעבוד בקביעות על חלק מטבלה בסיסית או על צירוף של חלקים של טבלאות בסיסיות. המבנה המופשט אינו מאוחסן בצורה פיסית בבסיס הנתונים אלא נבנה מתוך הטבלאות הבסיסיות בזמן הביצוע ואין לו קיום עצמאי בדומה לטבלאות הבסיסיות. המשתמש יכול לפעול בו־זמנית על מספר מבנים מופשטים כאלה, ולכן הם אינם מקבילים למושג תת־סכימה שבמודל הרשתי. תת־סכימה אחת מגדירה עבור המשתמש את כל סוגי הרשומות שהוא עוסק בהן.



תרשים 18 – 7: שלש הרמות של המערכת

טבלה בסיסית מוגדרת באמצעות פקודת CREATE-TABLE אשר כוללת את שם הטבלה, השדות והמפתחות שלה. בגמר ביצוע הפקודה נוצרת טבלה בסיסית ריקה חדשה שניתן להתחיל לרשום בה נתונים. לדוגמא נציג הגדרת טבלה חדשה SUPPLIER שבה שלשה שדות SUP-NO, SUP-NAME, CITY. כל השדות המוגדרים **כלא־ריקים** (NO-NULL) הם מפתחות של הטבלה ומשמשים לזיהוי הרשומה. בעת הזנת נתונים שדות אלו חייבים להכיל ערך כלשהו, נומרי או אלפביתי, בהתאם להגדרת השדה.

```
CREATE TABLE SUPPLIER (SUP-NO (CHAR (8), NO-NULL),
                       SUP-NAME (CHAR (30),
                       CITY (CHAR (20))
```

מבטלים טבלה מבסיס הנתונים באמצעות פקודות DROP TABLE, לדוגמא:

```
DROP TABLE SUPPLIER
```

קיימת אפשרות להרחיב טבלה בסיסית על ידי הוספת שדות חדשים (עמודות). לדוגמא, נוסיף את השדה "כתובת" לטבלת הספקים.

```
EXPAND TABLE SUPPLIER
        ADD FIELD SUP-ADDRESS (CHAR(40))
```

אפשר להגדיר אינדקסים לשדות (עמודות) של הטבלה כדי לשפר את זמן הביצוע של השאילתות והגישות לטבלה. למשתמש אין אפשרות להשתמש באינדקס בצורה ישירה ולכן קיומו או אי־קיומו של אינדקס אינה משפיעה על תוכניות היישום אלא רק על זמן הביצוע של הגישות לנתונים. ליצירת אינדקס SUP-INDEX לשדה SUP-NAME אשר בטבלה SUPPLIER נשתמש בהוראה הבאה:

```
CREATE INDEX SUP-INDEX ON SUPPLIER SUP-NAME
```

המבט של המשתמש על בסיס הנתונים, או על חלקו, מוגדר באמצעות פקודת DEFINE VIEW. זו הגדרה של **טבלה מדומה** אשר אינה קיימת באופן ממשי בבסיס הנתונים, אלא מוצגת מתוך טבלה בסיסית אחת או יותר בעת העיבוד.

DEFINE VIEW VALUED-ITEM	— שם הטבלה המדומה הכוללת
AS SELECT ITEM-NAME, PRICE	— שדות נבחרים מתוך
FROM ITEM	— הטבלה הבסיסית לפי
WHERE PRICE > 1000	— תנאי לבחירת הרשומות (שורות)

הטבלה המדומה VALUED-ITEM מורכבת מ־2 עמודות ושורותיה מכילות רק את הפריטים שמחירם עולה על 1000. לאחר הגדרת טבלה מדומה יכול המשתמש לפעול בה כאילו היתה טבלה קיימת באופן ממשי בבסיס הנתונים. הסבר נוסף על מבנה ההוראות ניתן בהמשך.

4.3 SEQUEL

SEQUEL היא שפה להגדרת הטבלאות והטיפול בהן במערכת R. השפה שואבת מספר רעיונות הן מאלגברה של היחסים והן מחשבן של יחסים. היא משתמשת בביטויים פשוטים בשפה האנגלית אשר מאפשרים רישום נוח ומובן של השאילתא על ידי המשתמש במערכת.

המבנה הבסיסי של פקודה בשפת SEQUEL:

SELECT target-list	— שדות (עמודות) נבחרים, מתוך
FROM relation	— שם הטבלה, לפי
WHERE logical-condition	— תנאים לבחירת הרשומות (שדות)
ORDER BY list	— את התוצאה למיין לפי...

נציג את השימוש בשפה באמצעות מספר דוגמאות המבוססות על תרשים 18—7.

א. דרושים פרטי הספק שמספרו S1.

(★ — כל העמודות)

```
SELECT    ★
FROM      SUP
WHERE     SUP-NO = 'S1'
```

ב. דרושים שם ומחיר של הפריטים שמחירם נמוך מ־100. את התוצאה יש למיין לפי שם הפריט.

```
SELECT      ITEM-NAME, PRICE
FROM        ITEM
WHERE       PRICE < 100
ORDER BY    ITEM-NAME
```

(אם לא צויין אחרת — מיון בסדר עולה)

ג. דרושה רשימת "מספרי הספק" של כל הספקים של פריט P_1.

```
SELECT    SUP-NO
FROM      ORD
WHERE     ITEM-NO = 'P1'
```

ד. דרושה רשימת שמות הספקים של פריט P_4.

```
SELECT    SUP-NAME
FROM      SUP
WHERE     SUP-NO =
          SELECT   SUP-NO
          FROM     ORD
          WHERE    ITEM-NO = 'P4'
```

מכיון שהשפה אינה מאפשרת בחירה ביותר מטבלה אחת, יש להשתמש בפקודות SELECT מוכלות (Nested). התוצאה של משפט SELECT בטבלה ORD (המשפט הפנימי) יפיק טבלה שתכלול רק את מספרי הספקים שמספקים פריט P_4. המשפט החיצוני הפועל על טבלה SUP יבחר מתוכה רק את שמות הספקים שנבחרו באמצעות המשפט הפנימי.

ה. דרוש לעדכן את מחיר פריט P_1 ל־100.

```
UPDATE    ITEM           — שם הטבלה
SET       PRICE=100      — המחיר החדש
WHERE     ITEM-NO='P1'   — בחירת הפריט לעדכון
```

ו. יש לבטל את רשומות הספקים של פריט P_6 בתנאי שמחיר הפריט נמוך מ־100.

★ — לציון כל העמודות בשורה

```
DELETE     ★
FROM       SUP
WHERE      SUP-NO =
           SELECT     SUP-NO
           FROM       ORD
           WHERE      ITEM-NO =
                      SELECT     ITEM-NO
                      FROM       ITEM
                      WHERE      ITEM-NO = 'P6' AND
                                 PRICE <100
```

בהוראה זו לפנינו שלש פקודות SELECT מוכלות. הפקודה ברמה הנמוכה, הפועלת על טבלה ITEM תבחר את הפריטים (מספרי פריט) אשר ממלאים תנאי כפול: מספרם P_6 ומחירם נמוך מ־100. רשימת מספרי פריט אלו תועבר למשפט הבא שפועל על טבלה ORD ובו תבחרנה רשומות המכילות את מספרי הספק של פריטים אלו. בשלב האחרון תבוצע הפעולה בטבלה SUP אשר תבטל את כל העמודות בשורה (סימן ★).

השימוש בכמה משפטים הפועלים על טבלאות אחדות נובע מכך שבטבלת הספקים (SUP) אין נתונים אשר יאפשרו את ביצוע הפעולה. נתונים אלו נמצאים בטבלאות אחרות. הקשר ביניהן הוא **דינמי** ונוצר בעת הדרישה האקראית לנתונים. זוהי אחת התכונות **העיקריות** של המודל הטבלאי — **אין בו קביעה מוקדמת של מסלולי גישה.**

במערכות מסחריות נכתבות הוראות SEQUEL בצורה פשוטה יותר ונוחה יותר מבחינתו של המשתמש. השוני הבולט לעומת הדוגמאות שלהלן הוא שאפשר לרשום יותר מטבלה אחת במשפט SELECT ואפשר לרשום תנאי שליפה מורכבים אשר מטפלים במספר טבלאות וביחס ביניהן. דוגמאות ראה בפרק SQL.

4.4 ניווט אוטומטי

מערכת R מאפשרת למשתמש גישה לנתונים וטיפול בהם מבלי שיצטרך לציין את מסלולי הגישה או את המבנה הפיסי של הנתונים. **בשיטה זו המשתמש אומר מה הוא רוצה ולא איך לעשות** כי המערכת בוחרת עבורו את הדרך האופטימלית להשגת התשובה. בתת־מערכת RDS קיימים אלגוריתמים שתפקידם לבחור באסטרטגית חיפוש אופטימלית לקבלת עלות מינימלית. האלגוריתם מתחשב באומדני צריכת זמן יע"מ ומספר פעולות קלט/פלט שיש לבצע. השירות הזה של המערכת גורם לכך שתוכניות היישום פשוטות יותר ואינן תלויות במבנה הנתונים.

4.5 אי־תלות בנתונים

שלש טכניקות פנימיות של מערכת R מאפשרות השגת ביצוע יעיל של הוראות השפה:

— קביעת מסלול גישה יעיל באמצעות אלגוריתם האופטימיזציה של RDS,
— הוספה או ביטול של מסלולי גישה לנתונים ללא כל השפעה על תוכנית היישום,
— הקבצת רשומות בשטח אחסון ליעול הגישה הסדרתית אליהן.

שלש השיטות האלה שקופות מבחינת המשתמש ועל כן מאפשרות לו אי־תלות בנתונים. למרות זאת יכולים הן המשתמש והן מנהל בסיס הנתונים להשפיע על יעילות המערכת. המשתמש יכול לבחור במבני שאילתא יעילים יותר ומנהל בסיס הנתונים יכול להגדיר יחידות ושטחי אחסון לשם שיפור הגישה. שניהם יוכלו להעזר למטרה זו בסטטיסטיקות התפעול המצטברות על־ידי המערכת בזמן עבודתה. שינויים שמנהל בסיס הנתונים יעשה בהגדרת שטחי אחסון לא ישפיעו על תוכניות היישום כל עוד לא יפגע בהגדרת הטבלאות.

5. סיכום

המודל הטבלאי מכיל מספר יתרונות על פני שני המודלים האחרים שהוצגו.

— **אי תלות בנתונים**. קיים שיפור ניכר במידת אי־התלות בנתונים וכל שינוי במבנה הפיסי או במודל הנתונים אינו משפיע על תוכניות היישום. המישק של המודל עם המשתמש אינו מכיל כל התייחסות למצביעים, שיטות גישה, מיקום בבסיס הנתונים וכדומה.

— **פשטות ונוחות שימוש**. מודל הנתונים משתמש רק בטבלאות דו־ממדיות ולכן הוא פשוט יותר מבחינת המשתמש לעומת מבני רשתות ועצים הירככיים. הטיפול בקשרים רב— רב־ערכיים הינו מיידי ואינו מחייב קשרים או מבנים מסייעים כמו בשני המודלים האחרים.

— **בסיס תיאורטי**. המודל הטבלאי מבוסס על תיאוריה מתמטית המאפשרת הגדרה מדוייקת וטובה של מבנה בסיס הנתונים והקשרים שבין הנתונים.

— **גמישות בגישה לנתונים**. המודל הטבלאי מציג מספר אופרטורים אלגבריים בעלי עוצמה רבה. אופרטורים כגון איחוד טבלאות, חיתוך טבלאות והיטל מאפשרים למשתמש מגוון רחב של פעולות לפתרון רוב הבעיות שעומדות בפניו.

למודל הטבלאי יתרונות חשובים אשר מייעדים אותו לשימוש בתחומים שונים ועל ידי משתמשים שונים באופן עצמאי ללא עזרת אנשי מקצוע בענ״א. הדבר בולט במיוחד למטרת הצגת שאילתות, הפקת דוחות מזדמנים ובניית תוכניות יישום מזדמנות. כולם מנצלים את התכונה החשובה של האפשרות לומר מה ולא איך, הפשטות של התקשורת אל המערכת וחשוב לא פחות — הגמישות הרבה בהגדרה ושינוי של הנתונים, השאילתות ותוכניות היישום. יתרונות אלו מכוונות את השימוש במודל בעיקר במערכות אינטראקטיביות.

עם זאת קיים קושי מסויים בשימוש במודל והוא הפעלתו במערכות שבהן נדרש זמן תגובה קצר לתנועות, בין אם הן מגיעות ברשת תקשורת או במערכות עיבוד באצווה. הדבר בולט במיוחד בבסיסי נתונים גדולים המכילים טבלאות רבות בגודל של מאות אלפים ומיליוני שורות כל אחת. מערכות אלו תשתמשנה במודל הירככי או רשתי. מכיון שהמודל ראה לפניו יעילות וגמישות בעבודת המשתמש הוא פגע ביעילות הביצוע במערכות גדולות.

ואולי נובע הדבר משוני מהותי בצורת העבודה במודלים השונים. המודל הטבלאי מקבל

משפט בשפת SEQUEL שהוא למעשה תוכנית עיבוד שלימה הפועלת על טבלאות ומפיקה טבלה כפלט. המודל ההירכי והמודל הרשתי מגיבים על הוראות קלט ופלט של תוכנית היישום וכל פעולתם — קריאה או כתיבה של רשומה אחת או יותר במסלול גישה מוגדר וקבוע. במודל הטבלאי הקשר בין הנתונים הוא דינמי בעת ביצוע המשפט של השפה והוא גמיש לחלוטין מבחינת המשתמש. במודלים האחרים הקשר בין הרשומות קבוע מראש ועל כן גם ניתן לאופטימיזציה ברמה גבוהה יותר.

היתרונות שבהפעלת המודל הטבלאי הם רבים ועל כן כבר קיימות מספר מערכות מסחריות המשתמשות בו. עבודת המחקר בעיצומה והיא תביא ללא ספק לשיפורים נוספים ולאפשרות להפעילו בתחומים נוספים.

6. שאלות

1. קרא את המשפטים הבאים העוסקים בפעולת היטל והשב "נכון/לא נכון".
   — בוחרת שורה בודדת מתוך טבלה.
   — בוחרת אף לא אחת, אחת או יותר שורות מתוך טבלה.
   — בוחרת את העמודה של המפתח הראשי.
   — בוחרת עמודה אחת מתוך טבלה.
   — בוחרת אחת או יותר עמודות מתוך טבלה.
   — בונה טבלה חדשה משתי טבלאות קיימות.
2. האם פעולת החיתוך מתאימה להגדרה הנ"ל?
3. אם גוזרים שתי טבלאות מתוך טבלה בצורה נכונה, האם ניתן לשחזר את הטבלה המקורית באמצעות אופרטור החיתוך?
4. מה משמעות המושג "דרגה של יחס"?
5. מהו ההבדל בין תחום (domain), ותכונה (attribute)?
6. מהו ההבדל בין מפתח ראשי לבין מפתחות משניים?

פרק ח

תכנון מודל הנתונים

(Data Model Design)

1. מבוא
2. תהליך תכנון בסיס הנתונים
3. תלות בין נתונים
4. תיקנון
5. צורות תקניות
6. איחוד מודלים של משתמש
7. שאלות
8. נספח – שיקולי שטח אחסנה במעבר לצורה תקנית שלישית

1. מבוא

מטרת תכנון מודל הנתונים הינו בנית מודל שיהווה ייצוג נכון ושלם של המציאות. בסיס הנתונים מייצג את הישויות ואת הארועים החשובים מנקודת מבט הארגון. מודל הנתונים הינו הבסיס לתכנון בסיס נתונים שיספק את צרכי עיבוד הנתונים של הארגון בצורה הטובה ביותר. חשוב לציין שכל ייצוג של המציאות הינו מטבע הדברים לא מושלם מאחר וזהו תיאור של המציאות ולא המציאות עצמה. הישויות והתכונות שנבחרו לייצוג המציאות הן הרלוונטיות ביותר מבחינת צרכי עיבוד הנתונים. לרוב מתעלמים מכל הישויות והתכונות שאינן רלוונטיות מבחינת הארגון.

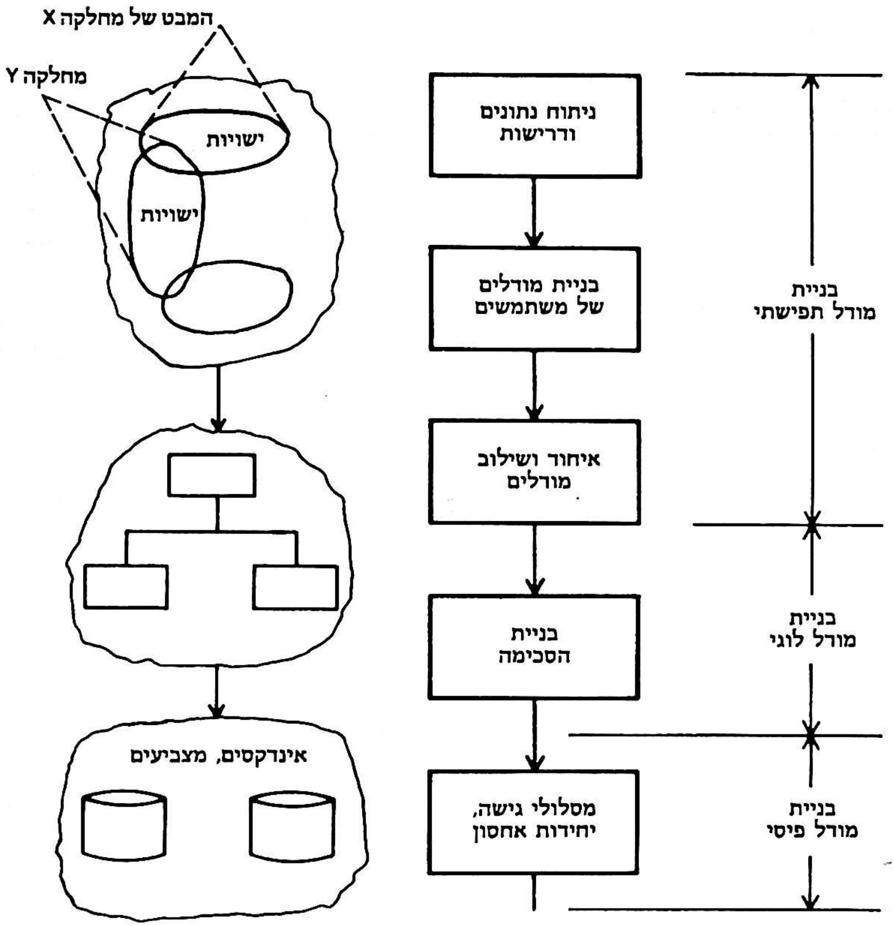
מודל הנתונים חייב לייצג את המציאות בצורה הנכונה והאמינה ביותר. מודל נתונים לא נכון מהווה סכנה מכיון שתהליכי קבלת ההחלטות המבוססים על מידע המתקבל מהמודל לא יהיו אמינים. לדוגמא המתכנן מחליט על קשר חד—רב־ערכי בין פריט וספק, כלומר פריט אחד מסופק על ידי מספר ספקים אבל לא להפך, בעוד שבמציאות מתקיים קשר רב—רב־ערכי בין שתי ישויות אלה. ברור שהמידע המופק מהמודל הזה יהיה לא אמין מאחר ואינו מייצג נכונה את המציאות. בעיה דומה יכולה להתעורר אם המתכנן מתעלם מחלק מהתכונות החשובות של הישות. לדוגמא יתכן שהתכונה "נפח" של פריט אינה מופיעה במודל. השמטת תכונה זו ממודל הנתונים תמנע הכנת תהליך אופטימיזציה ממוחשב לתכנון התובלה מאחר ונפח הפריט הינו אחד הפרמטרים החשובים.

תהליך תכנון מודל הנתונים הינו מורכב גם בגלל תלותו בזמן. ישויות ותכונות המיצגות את המציאות בצורה נכונה ושלימה היום, יתכן שלא תעשינה זאת מחר. נשתמש שוב בדוגמא של "נפח" הפריט. בזמן תכנון מודל הנתונים לא נאמר למתכנן שיש כוונה למחשב את תהליך תכנון העמסת המשאיות בסחורה ולכן המודל שהוא בנה היה שלם. לאחר הפעלת מערכת המידע נוצר הצורך בתכנון העמסת המשאיות בסחורה ואז התברר שאחד הפרמטרים החשובים לביצוע התכנון אינו מופיע במודל. זוהי דוגמא למודל שהינו שלם ונכון בנקודת זמן אחת ואינו נכון או שלם בנקודת זמן אחרת. עקרונית, המתכנן יכול להחליט שהמודל המתוכנן יכיל גם ישויות ותכונות נוספות מעבר לצרכי הארגון ברגע נתון. עליו לשקול את הגמישות ומהירות ההענות לצרכי מידע חדשים מול עלות האיסוף והעיבוד של נתונים שאינם רלוונטים לארגון כרגע.

בזמן תכנון מודל הנתונים עומדות בפני המתכנן מספר רב של אלטרנטיבות. חלק מהן טובות יותר וחלק מהן פחות טובות מכל מיני סיבות. לדוגמא אחת הבעיות השכיחות ביותר היא איזה תכונות לקבץ ביחד לקבלת רשומה או מבנה לוגי אחד. בדרך כלל קימות מספר צורות שונות בהן ניתן לקבץ אוסף של מאות ואלפי תכונות למודל נתונים אחד. חלק מהצורות מתגלה כטוב מבחינת עיבוד הנתונים בעוד חלק מהצורות ייצור בעיות רציניות בעתיד. כפי שנאמר, מודל הנתונים משתנה עם הזמן. בדרך כלל יש צורך להוסיף תכונות חדשות, ישויות חדשות, אפשרויות גישה לרשומות. אם שינויים אלה יחייבו שינויים כבדים במאות ואלפי תוכניות יישום הרי שעלות שמירת עדכניות ונכונות מודל הנתונים תהיה גבוהה למדי.

בפרק זה נתרכז בתהליך תכנון מודל הנתונים ונציג את אותן הנקודות שהן בעלות חשיבות בתהליך זה. אחד המושגים החשובים ביותר בתהליך בנית מודל הנתונים הינו מושג **התלות** (Dependency). התלות קובעת אילוצים מסוימים של הערכים שמודל הנתונים יכול לקבל.

2. תהליך תכנון בסיס הנתונים



תרשים 1 - 8: תהליך תכנון בסיס הנתונים

תהליך תכנון בסיס הנתונים מכיל שלשה שלבים עיקריים:
— תכנון המודל התפישתי,
— תכנון המודל לוגי,
— תכנון המודל הפיסי.

2.1 תכנון המודל התפישתי

אחד התפקידים החשובים ביותר של מנהל בסיס הנתונים (DBA) הוא **תכנון המודל התפישתי** (Conceptual Model) של הארגון. המודל חייב לייצג בצורה הטובה ביותר את הישויות והקשרים שביניהן באופן בלתי תלוי במערכת DBMS כלשהי או במערכת מחשב שבה הוא יופעל. בעת התכנון יש לעסוק בהבנת הנתונים והמידע הזורמים בארגון ולהתעלם מכל הנושאים של יישום המודל במערכת DBMS מסויימת.

תכנון המודל התפישתי כולל מספר שלבים עיקריים:

א. ניתוח נתונים ודרישות

בשלב זה יש לזהות את כל הנתונים ודרישות המידע בארגון. הזיהוי מתבצע באמצעות הטכניקות המקובלות של ניתוח מערכות כגון איסוף נתונים, ראיון, תצפיות וכדומה. מכיון שעורכים בשלב זה ניתוח דרישות של מספר יישומים שונים חשוב להשתמש בהקדרות אחידות ותקניות של הנתונים כדי למנוע סתירות וחוסר עקביות של המודל.

אחת הבעיות שבפניהן עומדים המתכננים הוא הצורך והקושי לאסוף נתונים אודות שימושים עתידיים של בסיס הנתונים. לאחר הפעלת בסיס הנתונים מגיעות בדרך כלל דרישות חדשות המתבטאות ביישומים חדשים, הוספת ישויות חדשות והוספת קשרים בין ישויות. איתור דרישות אלה בשלב תכנון המודל התפישתי יצמצם את הבעיות המתעוררות כאשר נדרשים שינויים ועדכונים במועד מאוחר יותר. מידע אודות נפחי נתונים ושיעור הפעילות והשימוש בהם יכול לעזור רבות בתכנון המודל הפיסי בשלב הבא.

בגמר תהליך ניתוח הדרישות חייב מנהל בסיס הנתונים להכיר את התמונה המלאה של צרכי עיבוד הנתונים בארגון. בהתבסס על הבנה זו עליו לזהות את הישויות הרלוונטיות, את התכונות של כל ישות ואת המפתחות של כל אחת מהן. בשלב זה עליו לדעת מספר נתונים פיסיים כגון מספר מופעים של ישות, גודל השדה הדרוש לייצוג תכונה מסויימת ועוד. לאחר קביעת הישויות ושמותיהן הוא מנתח את הקשרים ביניהן ובוחן את סוג ואופי הקשרים.

ב. בניית מודלים של משתמש

בעת ניתוח הנתונים והדרישות משתמשים במודל לייצוג הישויות והקשרים שביניהן. מכיון שבסיס הנתונים עתיד לשרת מספר רב של משתמשים, יש לבנות לכל משתמש את המודלים או את נקודת המבט שלו. כל אחד מאלו מהווה חלק מתוך המודל התפישתי הכולל ואמנם הם יאוחדו בשלב מאוחר יותר של התכנון. קיימת חפיפה בין שלב תכנון זה והשלב הקדם לו.

ג. איחוד מודלים

את כל המודלים של המשתמש המייצגים נקודות מבט שונות על אותם הנתונים יש לשלב ולאחד למודל תפישתי אחד כולל אשר יצג את כל הנתונים בארגון. קיימות שיטות שונות לאיחוד המודלים. אפשרות אחת היא להתחיל ברמה הנמוכה, כלומר ברמת התכונה או השדה של כל מודל ולבנות מהם מודל משולב. אפשרות אחרת היא להתחיל ברמת מודל המשתמש ולאחד את המודלים האלה למודל משולב. בגמר תהליך השילוב מתקבל המודל התפישתי של הארגון.

2.2 תכנון המודל הלוגי

המודל התפישתי שתוכנן בשלב הקודם אינו מתייחס למערכת DBMS מסויימת או למודל נתונים לוגי מוכר כמו מודל רשתי, הירכי או טבלאי. המודל התפישתי אינו נותן כל מידע על צורת אחסון הנתונים ועל שיטת השליפה שלהם ועל מנת ליישם את המערכת יש לבצע מיפוי שלו לאחד מהמודלים הלוגיים. קיימת גם אפשרות לפתח את המודל הלוגי באופן ישיר מתוך ניתוח הנתונים והדרישות אך עדיף לפתח תחילה את המודל התפישתי ואחריו — את המודל הלוגי.

המיפוי מהמודל התפישתי למודל הלוגי מתבצע על ידי מיפוי כל אחת מהישויות והקשרים למבנים המוכרים על־ידי מערכת DBMS מסויימת, כמו לדוגמא מיפוי לחבורות, לטבלאות או להירכיות. יתכן שבשלב זה יהיה צורך לבצע מספר שינויים כי לא תמיד המיפוי הוא ישיר ולעיתים צריך להוסיף רשומת קשר לייצוג קשר רב—רב־ערכי במודל CODASYL.

יש לשים לב לכך שבמודלים מסויימים יש לעסוק כבר בשלב זה בבעיות כמו מסלולי גישה השייכות מטבען לתכנון הפיסי. בתכנון המודל הטבלאי אין צורך לטפל בבעיות מסוג זה.

2.3 תכנון המודל הפיסי

מטרת תכנון המודל הפיסי היא להשיג את הביצועים הטובים ביותר בעלות הנמוכה ביותר. לכן עוסק תהליך התכנון בגורמים שמשפיעים על העלות ועל הביצועים כמו צריכת זמן יע"מ, מספר גישות קלט/פלט, נפח אחסון, טכניקות חיפוש ועוד. התכנון מורכב ממספר צעדים עיקריים:

א. **מיפוי הישויות למבנים פיסיים.** בשלב זה קובעים את המיפוי הפיסי של כל ישות שתיוצג במודל. במודל CODASYL למשל יש לקבוע את שטחי האחסון של כל סוג רשומה ואת הקירבה הפיסית שבין סוגי רשומות שונות.

ב. **מיפוי המבנים הפיסיים ליחידות אחסון.** בשלב זה מגדירים את יחידות ושטחי האחסון (גלילים ומסילות), שטחי גלישה, גודל מאגרי קלט/פלט, מספר רשומות לגוש (Blocking Factor), דחיסת נתונים, הצפנת נתונים וכדומה.

ג. **הגדרת מסלולי גישה.** בשלב האחרון קובעים את מסלול הגישה (Access Path) לכל רשומה בבסיס הנתונים. מסלול הגישה מגדיר את צורת הפנייה לבסיס הנתונים ואת הדרך לניווט עד לרשומת המטרה. הגישה לרשומה הראשונה בבסיס הנתונים הינה באמצעות אינדקסים או באמצעות תהליכי חישוב על המפתח והניווט נעשה באמצעות מצביעים כלשהם. במודל הטבלאי מסלולי הגישה הינם דינמיים ונקבעים על ידי המערכת בעת העיבוד ללא התערבות המשתמש, וללא הגדרה מוקדמת.

3. תלות בין נתונים

בזמן תכנון מודל הנתונים קובע המתכנן אילו ישויות ואילו תכונות של ישויות ייוצגו במודל ובאיזה אופן. ענין זה תקף בכל המודלים הפורמליים שהוצגו עד כה, ההירכי, הרשתי והטבלאי. המושגים המוגדרים בהמשך, השיקולים והבעיות הנדונים אינם תלויים במודל זה או אחר ועל כן המתכנן חייב לעסוק בהם ללא תלות במערכת ניהול בסיסי הנתונים ליישום המודל.

העיסוק בנושא החל עם פיתוח המודל הטבלאי הכולל בסיס מתמטי מפורט. בהמשך נשתמש במספר דוגמאות המבוססות על המודל הטבלאי בעיקר בגלל פשטותו ונוחות הטיפול בו. הצגת הנושא הינה אינטואיטיבית ולא מתמטית פורמלית. הקורא המעוניין בהוכחות הפורמליות מופנה לספרות העניפה בנושא המופיעה ברשימה הביבליוגרפית.

3.1 תלות פונקציונלית (Functional Dependency)

תלות פונקציונלית קיימת בין תכונה B ביחס R לבין תכונה A אם בכל נקודת זמן כל ערך של A קשור לערך אחד בלבד של B. במילים אחרות, תכונה A מזהה את תכונה B בצורה ש־B הינה פונקציה של A.

F=[<a,b>| a מתוך A, b מתוך B]

התלות הפונקציונלית קובעת שקיימת פונקציה F שהינה אוסף מסודר של זוגות של ערכים <a,b> כך שהערך a נלקח מתוך תחום הערכים של התכונה A והערך b נלקח מתוך תחום הערכים של התכונה B. בכל נקודת זמן עבור כל ערך a קיים ערך b אחד לכל היותר.

נשתמש ביחס "סטודנטים" המוצג בתרשים 2—8 להמחשת המושג "תלות פונקציונלית":

סטודנטים (מספר־סטודנט, שם־סטודנט, כתובת־מגורים, שנת־לידה, ארץ־לידה)

סטודנטים

מספר־סטודנט	שם־סטודנט	כתובת־מגורים	שנת־לידה	ארץ־לידה
221	דוידי משה	עגנון 9, חיפה	1945	ישראל
125	שרון איתן	ביאליק 10, נתניה	1952	ישראל
345	דוידי משה	הרצל 5, ת־א	1960	צרפת
904	שרון אילת	אשכול 12, נתניה	1959	אלגיר
111	כהן איל	עגנון 9, חיפה	1962	צרפת
212	לוי אהרון	הרואה 10, רמת־גן	1959	גרמניה

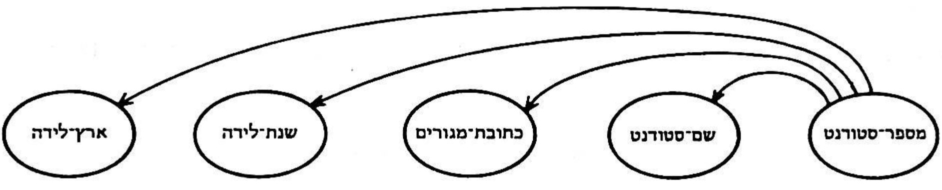
תרשים 2 – 8: יחס "סטודנטים"

המפתח של היחס הוא התכונה "מספר־סטודנט" מכיון שכל שאר התכונות אינן מזהות בצורה חד־משמעית את השורות בטבלה. לכל התכונות המופיעות ביחס "סטודנטים" תלות

פונקציונלית בתכונה "מספר־סטודנט" ובכך כל מספר סטודנט מזהה שם־סטודנט אחד בלבד או כתובת־מגורים אחת בלבד. בתרשים 2—8, מספר סטודנט 111 מזהה סטודנט ששמו כהן איל שגר בחיפה ברחוב עגנון 9.

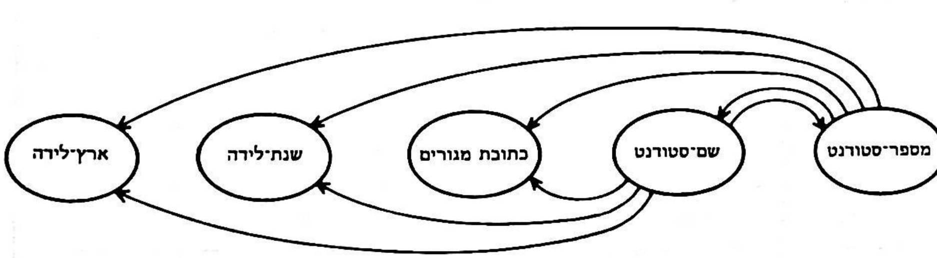
בכיוון ההפוך לא קיימת תלות פונקציונלית כי לדוגמא התכונה "כתובת־מגורים" אינה מגדירה מספר סטודנט אחד בלבד. בתרשים 2—8, בכתובת עגנון 9 חיפה גרים שני סטודנטים שונים: כהן אלי ודוידי משה. בדומה, גם התכונות "שנת־לידה", "ארץ־לידה" או "שם־סטודנט", אינן מגדירות מספר סטודנט אחד בלבד: בשנת לידה 1959 נולדו שני הסטודנטים לוי אהרון ושרון אילת.

אפשר להציג באופן גרפי את התלות הפונקציונלית בתוך היחס כפי שמודגם בתרשים 3—8.



תרשים 3 - 8: הצגה גרפית לתלות פונקציונלית

אם היינו מניחים שלא יתכנו שני סטודנטים בעלי שם זהה, היחס "סטודנטים" היה מכיל שני מפתחות אפשריים: מספר־סטודנט ושם־סטודנט. כל אחת מתכונות אלו תלוייה בתלות פונקציונלית בתכונה השניה, ולכל שאר התכונות תלות פונקציונלית בשתי תכונות אלו. מצב זה מוצג בתרשים 4—8.



תרשים 4 - 8: תלות פונקציונלית כפולה

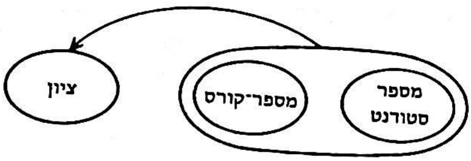
לתכונה עשויה להיות תלות פונקציונלית בקבוצה של תכונות ולא רק בתכונה אחת אחרת. דוגמא לכך יכולה להיות תלות של תכונה במפתח מורכב. נשתמש ביחס "ציון־בקורס" שבתרשים 5—8 להדגמת תלות פונקציונלית בקבוצת תכונות.

ציון־בקורס (<u>מספר־סטודנט, מספר־קורס</u>, ציון)

לתכונה "ציון" תלות פונקציונלית בשתי התכונות "מספר־סטודנט", ו"מספר־קורס" כאחד. הציון אינו תלוי במספר־סטודנט או במספר קורס בנפרד אלא בשילוב שביניהם. זוהי "תלות פונקציונלית שלימה" אשר תוסבר בהמשך.

ציון־בקורס

מספר־סטודנט	מספר־קורס	ציון
221	113	80
221	117	95
345	113	75
111	124	80



תרשים 5 – 8: יחס "ציון־בקורס"

אם קיימות ביחס R תכונות שתלויות תלות פונקציונלית בתכונות אחרות של היחס, ניתן לפצל אותו למספר יחסים במימד קטן יותר מהמימד של R. לדוגמא ניתן לפצל את היחס "סטודנטים" שמופיע בתרשים 2—8 למספר יחסים מכיון שלכל התכונות תלות פונקציונלית במפתח. יחסים אלה מודגמים בתרשים 6—8. הפיצול אינו גורם לאבדן נתונים כי אם נאחד את היחסים המפוצלים נוכל לשחזר את היחס המקורי. מן הראוי לשים לב לכך שהפיצול גרם למידה רבה של כפילות נתונים.

R1

מספר־סטודנט	שם־סטודנט
221	דווידי משה
125	שרון איתן
345	דווידי משה
904	שרון אילת
111	כהן איל
212	לוי אהרון

R2

מספר־סטודנט	כתובת־מגורים
221	עגנון 9, חיפה
125	ביאליק 10, נתניה
345	הרצל 5, ת״א
904	אשכול 12, נתניה
111	עגנון 9, חיפה
212	הרואה 10, רמת־גן

R3

מספר־סטודנט	שנת־לידה
221	1945
125	1952
345	1960
904	1959
111	1962
212	1959

R4

מספר־סטודנט	ארץ־לידה
221	ישראל
125	ישראל
345	צרפת
904	אלגיר
111	צרפת
212	גרמניה

תרשים 6 – 8: פיצול יחסים

התלות הפונקציונלית מהווה חלק מהבנת **המשמעות של הנתונים** (Data Semantics). התלות הפונקציונלית בין כתובת המגורים למשל לבין מספר הסטודנט מייצגת את **העובדה** שלכל סטודנט מוגדרת בדיוק כתובת מגורים אחת. זהו למעשה אילוץ הקיים במציאות ובסיס הנתונים צריך לייצג אותו. האילוץ צריך לבוא לידי ביטוי במודל הנתונים כך שמערכת DBMS תוכל לודא את קיומו.

3.2 תלות פונקציונלית שלימה (Full Functional Dependency)

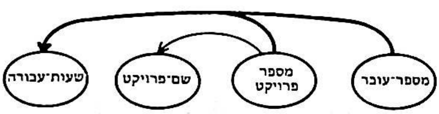
תלות פונקציונלית שלימה קיימת בין תכונה C ביחס R לבין התכונות A וְ־B אם היא תלוייה בשתיהן כאחת ולא בכל אחת מהן לחוד. נשתמש ביחס "שעות פרויקט" שבתרשים 7—8 להסבר המושג תלות פונקציונלית שלימה:

שעות־פרוייקט (<u>מספר־עובד, מספר־פרויקט</u>, שם־פרויקט, שעות־עבודה)

התכונה "שעות־עבודה" מייצגת את מספר שעות העבודה המצטברות שעובד מסויים השקיע בפרויקט כלשהו, ויש לה תלות פונקציונלית שלימה במפתח המורכב מתכונות "מספר־עובד" ו"מספר־פרויקט". לעומת זאת התכונה "שם־פרויקט" אינה תלויה במפתח המורכב אלא רק בחלק ממנו, בתכונה "מספר־פרויקט" בלבד.

שעות־פרויקט

מספר־עובד	מספר־פרויקט	שם־פרויקט	שעות־עבודה
125	124	כפיר	300
125	157	זברה	345
341	132	לביא	210
395	124	כפיר	125
412	157	זברה	140
312	157	זברה	300



תרשים 7 – 8: תלות פונקציונלית שלימה

אם לתכונה מסויימת יש תלות פונקציונלית שלימה בקבוצת תכונות ומפצלים את היחס שהיא נמצאת בו ליחסים במימד קטן יותר, היחס החדש **חייב** להכיל את המפתח המורכב ואין אפשרות לבצע פיצול נוסף. פיצול היחס "שעות־פרוייקט" לשני יחסים במימד קטן יותר יצר יחס חדש "שעות־שהושקעו" אשר מכיל את התכונה "שעות־עבודה" ואת המפתח המורכב כולו. ביחס "פרויקטים" קיימת תלות פונקציונלית רגילה.

שעות־שהושקעו (<u>מספר־עובד, מספר־פרויקט</u>, שעות־עבודה)

פרויקטים (<u>מספר־פרויקט</u>, שם־פרויקט)

3.3 תלות עוברת (Transitive Dependency)

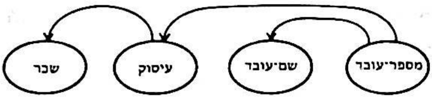
תלות עוברת קיימת בין תכונה C ביחס R לבין תכונה A אם לתכונה C תלות פונקציונלית בתכונה B ולתכונה B תלות פונקציונלית בתכונה A. במקרה זה קיימת **תלות עקיפה** בין תכונה C לבין תכונה A.

נשתמש ביחס "שכר" שבתרשים 8—8 להסבר:

שכר (<u>מספר־עובד</u>, שם־עובד, עיסוק, שכר)

שכר

מספר־עובד	שם־עובד	עיסוק	שכר
125	משה גל	מתכנת	15,000
341	דן עמית	מנתח־מערכות	20,000
395	רן שרוני	מתכנת	15,000
412	כהן אמנון	מנתח־מערכות	20,000
312	יצחקי דוד	מתכנת	15,000



תרשים 8 – 8: תלות עוברת

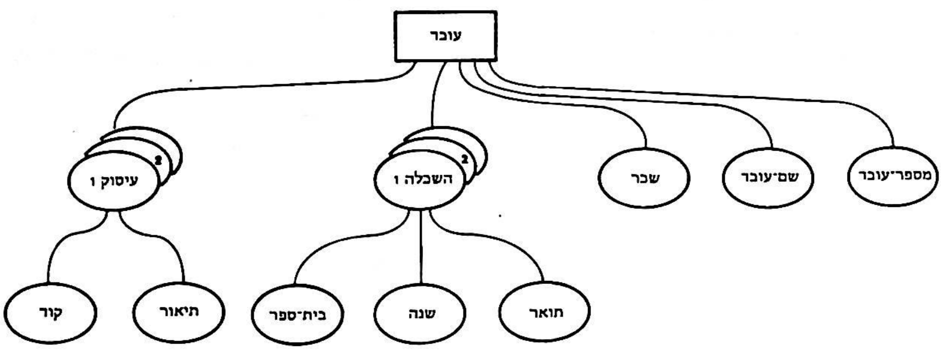
לעיסוק העובד תלות פונקציונלית במפתח "מספר־עובד", שמשמעה: לכל עובד יש בנקודת זמן עיסוק אחד. מאידך, לשכר אין תלות פונקציונלית במספר העובד כי הוא נקבע על פי העיסוק של העובד, כפי שגם מוצג בתרשים 8—8. בין השכר לבין מספר העובד קיימת תלות עוברת ולא תלות פונקציונלית.

4. תיקנון

מטרת תהליך **התיקנון** (Normalization) הינה בניית מודל נתונים יעיל ואמין מבחינת איכות הנתונים המופיעים בו. מושג התיקנון קשור לפיתוח המודל הטבלאי כשם שקשורים בו נושאים אחרים בתחום מערכות בסיסי נתונים. בעת הצגת המודל הטבלאי הגדיר E.F. Codd שלש רמות תיקנון שונות, ובשלב מאוחר יותר הגדירו חוקרים אחרים רמות נוספות.

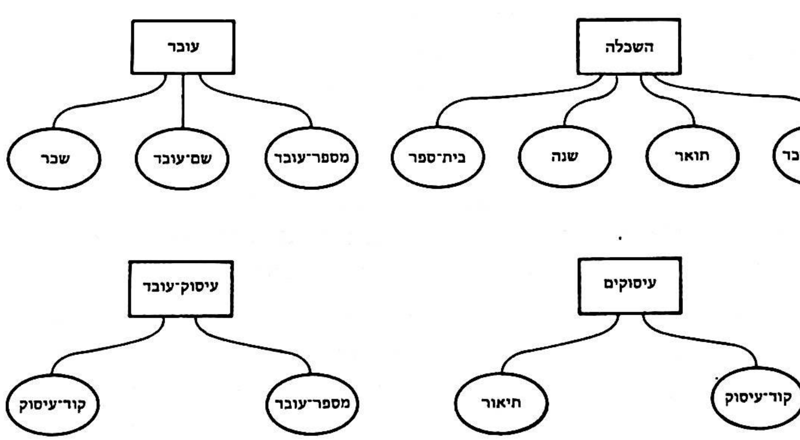
תהליך התיקנון הינו תהליך אלגוריתמי שמחליף אוסף של יחסים לא מתוקננים באוסף אחר של יחסים תואמים, בעלי מבנה פשוט ומסודר יותר. זהו **תהליך הפיך** (Reversible Process) אשר מבטיח שלא יאבדו במהלכו כל נתונים שהם. אפשר להגדיר את התהליך גם בצורה הבאה:

תהליך התיקנון הינו הפיכת מבנה יחס מורכב למבנה פשוט יותר באמצעות ביטול קשרי התלות בין התכונות השונות, ללא איבוד נתונים.



תרשים 9 - 8: מבנה נתונים מורכב

תרשים 9—8 מציג מבנה נתונים מורכב אשר התיקנון שלו יוביל לבניית ארבעה מבנים פשוטים יותר המודגמים בתרשים 10—8.



תרשים 10 - 8: מבנים מתוקננים

4.1 אנומליות בפעולות אחסנה (Anomalies in Storage Operations)

להצגת הבעיות הנובעות מטיפול ביחסים לא מתוקננים נשתמש בדוגמא של היחס "עובדים־במפעל" אשר כולל את התכונות הבאות: מספר זהות של העובד, מספר הפרויקט שבו הוא מועסק, שם העובד, שם הפרויקט, תאריך שהתחיל את העבודה בפרויקט ותאריך שסיים את העבודה בפרויקט. המפתח הראשי מורכב משתי התכונות "מספר־זהות" ו"מספר־פרויקט" מכיון שאותו עובד עשוי לעסוק במספר פרויקטים ובפרויקט אחד עשויים לעסוק מספר עובדים:

עובדים־במפעל (מספר־זהות, מספר־פרויקט, שם־עובד, שם־פרויקט, תאריך־התחלה, תאריך־סיום)

עובדים־במפעל

מספר־זהות	מספר־פרויקט	שם־עובד	שם־פרויקט	תאריך־התחלה	תאריך־סיום
62456712	124	משה גל	כפיר	12.1.78	20.5.83
62456712	157	משה גל	שרשרת	12.1.80	20.5.83
62456712	132	משה גל	לביא	1.12.79	13.8.82
82152462	124	דן עמית	כפיר	12.2.78	20.3.83
75246214	159	רן שרוני	זברה	12.3.78	20.5.83

תרשים 11 – 8: יחס לא מתוקנן עם מפתח מורכב

"אנומליות" הן שם כולל לבעיות הנגרמות בעת הקמת מודל נתונים. מקובל לזהות את האנומליות הבאות אשר נסביר אותן באמצעות תרשים 11—8:

א. **אנומלית ההוספה** (Insertion Anomaly). לא ניתן להוסיף נתונים אודות פרוייקט חדש אלא אם ידוע לפחות עובד אחד שמועסק בו. הסיבה לכך היא שהמפתח מורכב משני שדות מספר־זהות ומספר־פרויקט, ולכן שניהם חייבים להיות מוגדרים. בדומה לא ניתן להוסיף נתונים אודות עובד חדש אלא אם ידוע לפחות פרויקט אחד שבו הוא מועסק. כאשר מוסיפים נתונים על עובד כלשהו חייבים לציין נתונים נוספים כמו נתוני הפרויקט למשל, אם זהו העובד הראשון בפרויקט למרות שלא זו היתה הכוונה בעת הוספת העובד. בתרשים 11—8 פתיחת הרשומה של רן שרוני גורמת את הוספת הנתונים על פרויקט מספר 159.

ב. **אנומלית הביטול** (Deletion Anomaly). ביטול נתונים של העובד עלולה לגרום לאובדן נתונים של הפרויקט. ביטול הרשומה של רן שרוני למשל תגרום את אובדן הנתונים של שם פרויקט זברה. בדומה, עם סיום פרויקט כלשהו, ביטול הנתונים שלו עלול לגרום את אובדן הנתונים על העובד. עם ביטול הרשומות המתייחסות לפרויקט כפיר, למשל, נאבד נתונים על העובד דן עמית מכיון שהוא עובד בפרויקט זה בלבד.

ג. **אנומלית העדכון** (Modify Anomaly). השינוי של שם פרויקט כלשהו מחייב עדכון מספר רב של רשומות מכיון שנתון זה מופיע במספר רשומות. אם המשתמש אינו ער לכך יתכן שהוא יעדכן רק את הרשומה הראשונה בלבד וייצור בכך מצב של חוסר אמינות כי לאותו מספר־פרויקט יהיו מספר שמות שונים. ניתן להוסיף לטבלה שורה עם התוכן הבא:

<51267238, 124, אבי רונן, ערבה, 12.1.78, 22.5.83>

הוספת שורה זו תגרום מצב שבו לפרויקט 124 יהיו שני שמות שונים בבסיס הנתונים — "כפיר" ו"ערבה" ובכך תיווצר סתירה למציאות המיוצגת במודל: במציאות שם פרויקט 124 הוא "כפיר". זוהי דוגמא לחוסר האמינות בבסיס הנתונים הנובעת מהופעת התכונה "שם־פרויקט" מספר רב של פעמים. צורת הצגה זו גורמת לבזבוז בשטחי אחסון ולמאמצים מיותרים בעת עדכון בסיס הנתונים.

לסיכום, מטרות תהליך התיקנון הן:

— לאפשר ייצוג כל היחסים בבסיס הנתונים,
— לאפשר שימוש באלגוריתמים יעילים לאחזור הנתונים אשר מבוססים על אוסף של אופרטורים פשוטים,
— לשחרר את היחסים המיוצגים בבסיס הנתונים מהאנומליות הבלתי רצויות בזמן ביצוע פעולות הוספה, עדכון וביטול,
— לצמצם את הצורך לבצע שינויים במבנה היחסים לאחר הוספת תכונות או יחסים חדשים.

תהליך התיקנון מבוסס על **הבנת התלות** בין הנתונים המיוצגים בבסיס הנתונים, שמשמעה, זיהוי איזה נתון תלוי באיזה נתון ובאיזו צורה.

בהמשך נדון בצורות תקניות של יחסים.

5. צורות תקניות

5.1 צורה תקנית ראשונה (1NF — First Normal Form)

יחס נמצא בצורה תקנית ראשונה כאשר כל התכונות ביחס מבוססות על תחום פשוט. מקובל לומר שהתכונות הן **אטומיות** שאינן ניתנות לחלוקה נוספת. המשמעות היא, שבטבלה המייצגת את היחס יש ערך אחד בדיוק בכל חיתוך שורה—עמודה, ולא אוסף של מספר ערכים. ערכים לא מוגדרים מותרים רק עבור תכונות שאינן משמשות כמפתחות.

עובדים

מספר-עובד	שם-עובד	ילדים	מנהל-של...
125	משה גל	אמיר	דן עמית
		רונן	רן שרוני
			כהן אמנון
341	דן עמית	אייל	-
		שרית	-
		אילה	-
395	רן שרוני	-	-
412	כהן אמנון	צבי	-
		דינה	-

תרשים 12 - 8: יחס לא מתוקנן

בדוגמא שבתרשים 12—8 התכונות "ילדים" ו־"מנהל־של" אינן תכונות פשוטות. לכל עובד יתכנו מספר ילדים ועובד אחד יכול להיות מנהל של מספר עובדים אחרים. מספר המופעים של כל תכונה כזאת הוא משתנה ובלתי ידוע. לדוגמא נציג שורה אחת מתוך היחס "עובדים":

〈 125, משה גל, אמיר, דן עמית 〉
רונן רן שרוני
כהן אמנון

כפי שניתן לראות, לעובד בשם משה גל שני ילדים והוא המנהל של שלשה עובדים אחרים. מאחר וכל ערך שמופיע ביחס הזה אינו אטומי נאמר שהיחס אינו נמצא בצורה תקנית ראשונה. ניתן לעבור לצורה תקנית ראשונה על ידי סילוק המופיעים המשתנים של הערכים בתוך היחס. פעולה זו אפשר לבצע בשתי דרכים:

א. נניח שמספר הילדים המירבי לעובד הוא עשרה. על סמך הנחה זו נבנה שני יחסים:

עובד (מספר־עובד, שם־עובד, ילד־1, ילד־2, ..., ילד־10)
מנהל (מספר־עובד־מנהל, מספר־עובד־כפוף)

ביחס הראשון פרסנו את התכונה "ילדים" למספר תכונות שכל אחת מהן יכולה לקבל ערך אטומי. מאחר ואין אפשרות להניח מה מספר העובדים הכפופים למנהל מסויים בנינו יחס נוסף "מנהל" שמקשר בין מספר העובד של המנהל לבין מספר העובד הכפוף לו.

עובד

מספר־עובד	שם־עובד	ילד־1	ילד־2	ילד־3	....	ילד־10
125	משה גל	אמיר	רונן			
341	דן עמית	אייל	שרית	אילה		
395	רן שרוני	-	-	-		
412	כהן אמנון	צבי	דינה			

מנהל

מספר־עובד מנהל	מספר־עובד כפוף
125	341
125	395
125	412

תרשים 13 – 8: יחס בצורה תקנית ראשונה - אפשרות א'

ב. אם ההנחה בדבר מספר הילדים של העובד לא נראית סבירה ניתן לבנות יחס נוסף "ילדים" הקושר את העובד עם הילדים שלו.

עובד

מספר־עובד	שם־עובד
125	משה גל
341	דן עמית
395	רן שרוני
412	כהן אמנון

ילדים

מספר־עובד	שם־ילד
125	אמיר
125	רונן
341	אייל
341	שרית
341	אילה
412	צבי
412	דינה

מנהל

מספר־עובד מנהל	מספר־עובד כפוף
125	341
125	395
125	412

תרשים 14 – 8: יחס בצורה תקנית ראשונה - אפשרות ב'

המעבר לצורה תקנית ראשונה אינו גורם לאבדן נתונים אלא לשינוי המבנה של היחסים שביניהם. התיקון גרם לסילוק הערכים בעלי מופע משתנה ולבניית יחסים תואמים חדשים שמכילים רק ערכים אטומיים. היתרון של יחסים מסוג 1NF לעומת יחסים לא מתוקננים הינו בפשטות הפעולות על היחסים האלו. כתוצאה מכך גם תוכניות היישום המטפלות במבנים אלה פשוטות יותר. עם זאת יש כאן מורכבות מסויימות הנובעת מכך שהתכונה "מספר עובד" נמצאת במספר יחסים שונים. לא די להכיר רק את היחס הלא מתוקנן "עובד" אלא יש להכיר את שלושת היחסים "עובד", "ילדים" ו"מנהל".

כל סוגי התלות בין התכונות יכולים להתקיים ביחס בצורתו התקנית הראשונה. הדרישה היחידה היא שהוא יכיל ערכים אטומיים בלבד, כלומר ערכים פשוטים.

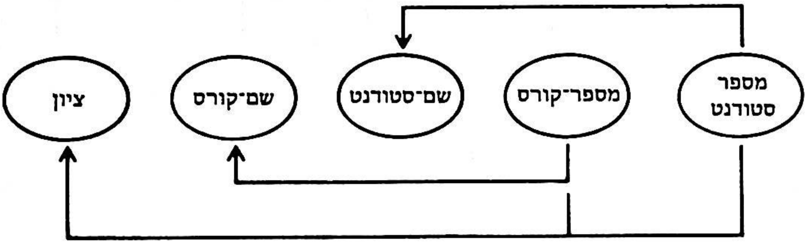
5.2 צורה תקנית שניה (2NF—Second Normal Form)

יחס נמצא בצורה תקנית שניה כאשר לכל התכונות ביחס תלות פונקציונלית מלאה בכל המפתחות האפשריים ביחס. המעבר מצורה תקנית ראשונה לצורה תקנית שניה כרוך בסילוק תכונות בעלות תלות חלקית במפתחות האפשריים והוא הכרחי רק אם המפתח מכיל מספר תכונות. כתוצאה מכך יתכן שתהיינה תכונות שתלויות רק בחלק מהמפתח. כאשר המפתח מכיל תכונה אחת בלבד לא קיימת תלות חלקית. יחס 2NF יכול להכיל תלות עוברת.

היחס "קורסים" מוצג בתרשים 15—8:

קורסים (מספר־סטודנט, מספר־קורס, שם־סטודנט, שם־קורס, ציון)

ליחס זה מפתח אפשרי אחד: "מספר־סטודנט" ו"מספר־קורס" כי סטודנט אחד יכול להשתתף במספר קורסים ובקורס אחד משתתפים מספר סטודנטים. ניתן לראות שלתכונה "שם־סטודנט" אין תלות פונקציונלית במפתח כולו אלא במספר הסטודנט בלבד. תלות חלקית קיימת בין "שם־קורס" לבין המפתח כי הוא תלוי ב"מספר־קורס" בלבד. רק ל"ציון" יש תלות פונקציונלית מלאה במפתח. מכיון שלא לכל התכונות ביחס "קורסים" תלות פונקציונלית מלאה במפתח, נאמר שהיחס אינו נמצא בצורה תקנית שניה.



קורסים

מספר־סטודנט	מספר־קורס	שם־סטודנט	שם־קורס	ציון
155	17	אבי	אלגברה	95
155	19	אבי	תכנות	100
155	24	אבי	כלכלה	92
162	19	רוני	תכנות	74
164	24	יצחק	כלכלה	75
164	18	יצחק	הסטוריה	82

תרשים 15 – 8: יחס בצורה תקנית ראשונה

ביחס הנמצא בצורה התקנית הראשונה אפשר להבחין במספר תופעות לואי (אנומליות) שהוסברו קודם:

— לא ניתן להוסיף נתונים על ציון של סטודנט אלא אם יודעים את שם הקורס שבו הוא משתתף,
— לא ניתן להוסיף נתונים על קורס אלא אם ידוע לפחות סטודנט אחד שמשתתף בו,
— שם הקורס ושם הסטודנט מופיעים מספר רב של פעמים בבסיס הנתונים ויש סיכון של חוסר עקביות.

את כל תופעות הלואי האלה ניתן לסלק אם היחס יועבר לצורה התקנית השניה. העברה זו כרוכה בפיצולו לשלשה יחסים נפרדים:

קורסים (מספר־קורס, שם־קורס)
סטודנטים (מספר־סטודנט, שם־סטודנט)
ציונים (מספר־סטודנט, מספר־קורס, ציון)

מספר קורס	שם־קורס
17	אלגברה
18	הסטוריה
19	תכנות
24	כלכלה

מספר־סטודנט	שם־סטודנט
155	אבי
162	רוני
164	יצחק

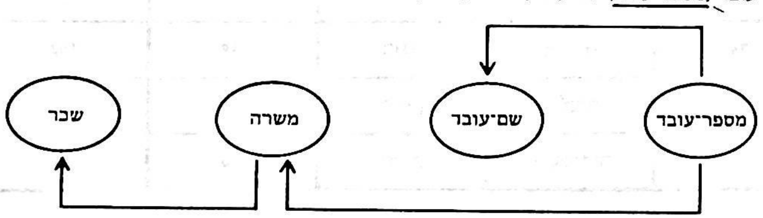
מספר־סטודנט	מספר־קורס	ציון
155	17	100
155	24	92
162	19	74
164	24	75
164	18	82

תרשים 16 – 8: יחס בצורה תקנית שנייה

5.3 צורה תקנית שלישית (3NF—Third Normal Form)

יחס נמצא בצורה תקנית שלישית כאשר לכל התכונות תלות פונקציונלית מלאה במפתח. יחס זה מוצג כמו יחס בצורה תקנית שנייה אך לכל התכונות שלו אין תלות עוברת במפתחות האפשריים שלו.

עובד (מספר־עובד, שם־עובד, משרה, שכר)



תרשים 17 – 8: יחס המכיל תלות עוברת

ביחס "עובד" בתרשים 17—8 קיימת תלות עוברת בין המשרה והשכר כי השכר נקבע על־פי המשרה של העובד ולא על־פי מספר־העובד או שם־העובד. ביטול תלות עוברת זו אפשרי על ידי בניית יחס חדש "משרות" שמכיל את השכר המשולם בהתאם למשרה.

עובד (מספר־עובד, שם־עובד, משרה)
משרות (משרה, שכר)

המעבר מצורה תקנית שניה לצורה תקנית שלישית מונע את תופעות הלואי הכרוכות בעדכון בסיס הנתונים ואת הכפילויות העלולות להיות במודל הנתונים. כאשר יחס נמצא בצורה תקנית שניה יגרום שינוי השכר למשרה מסויימת את סריקת כל בסיס הנתונים ועדכון כל רשומות העובד המתאימות למשרה זו. בצורה התקנית השלישית הקשר בין המשרה לשכר מופיע פעם אחת בלבד ולכן ביצוע השינוי הוא פשוט והאמינות של בסיס הנתונים נשמרת.

הצגת היחס בצורה תקנית שלישית אינה בהכרח הטובה ביותר. קיימים מספר מקרים שבהם עדיף לאחסן נתונים בצורה תקנית ראשונה או שניה, כמו למשל מתוך שיקולים של יעילות ביצועים. בדוגמא הקודמת יש צורך בגישה אחת בלבד לאחזור נתוני עובד כאשר היחס מוצג בצורה תקנית שניה לעומת שתי גישות אשר דרושות ביחס בצורה תקנית שלישית. על המתכנן לשקול את יעילות ופשטות אחזור הנתונים ביחס מול תופעות הלואי אשר קיימות ביחסים שאינם בצורה תקנית שלישית.

5.4 צורה תקנית שלישית משופרת (BCNF—Boyce/Codd Normal Form)

ההגדרה המקורית של הצורה התקנית השלישית על־ידי Codd לא היתה שלימה ולא נתנה תשובה למקרים מסויימים. צורה שלישית משופרת פותחה על־ידי Boyce וְ־Codd, והיא נקראת על שמם — BCNF, כדי להבדילה מהצורה התקנית השלישית הקודמת. הגדרת BCNF אינה משתמשת בצורה ישירה במושגים של תלות פונקציונלית שלימה ותלות עוברת ואינה מתבססת על הצורות התקניות הראשונה והשניה. היא משתמשת במושג **תכונה מגדירה** (Determinant) לתיאור תכונה אשר תכונות אחרות תלויות בה.

יחס נמצא בצורה התקנית השלישית המשופרת BCNF אם ורק אם כל תכונה מגדירה היא מפתח אפשרי. הגדרה זו מתייחסת לכל המפתחות האפשריים ולא רק למפתח הראשי. כל יחס שנמצא בצורה השלישית המשופרת הוא בהכרח גם בצורתו השלישית הקודמת אך לא להיפך.

הצורך בהגדרה החדשה נובע מהטיפול הלא מספק של יחס 3NF במקרים של ריבוי מפתחות וחפיפה ביניהם. חפיפה בין שני מפתחות אפשריים תיתכן רק אם הן מכילים שתי תכונות או יותר.

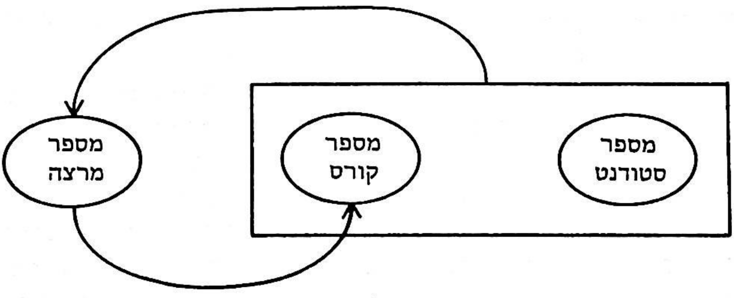
נתון יחס "קורסים" אשר נמצא בצורה תקנית שלישית וכולל מספר אילוצים:

— קורס יכול להנתן על ידי מספר מרצים שונים,
— כל מרצה מלמד בקורס אחד בלבד,
— בכל קורס הסטודנט לומד אצל מרצה אחד בלבד.

קורסים (מספר־סטודנט, מספר־קורס, מספר־מרצה)

האילוצים יוצרים את התלויות הבאות בהתאמה:

— אין תלות פונקציונלית בין מספר הקורס למספר המרצה,
— קיימת תלות פונקציונלית בין מספר הקורס לבין מספר המרצה,
— קיימת תלות פונקציונלית בין מספר המרצה לבין התכונה המורכבת מספר־סטודנט ומספר־קורס.



תרשים 18 - 8: תלויות ביחס "קורסים"

בתרשים 8—18 מוצגות התלויות השונות ביחס והן מתארות מצב שבו יש ליחס שני מפתחות חופפים. המפתח הראשון הוא צירוף של מספר־סטודנט ומספר־קורס והמפתח השני הוא צירוף של מספר־סטודנט ומספר־מרצה. יחס זה נמצא בצורה התקנית השלישית אבל לא בצורת BCNF מכיון שמספר־מרצה הוא תכונה מגדירה אבל לא מפתח אפשרי. יחס זה סובל מהאנומליות שצויינו קודם. אם נבטל את הקשר בין סטודנט לקורס נאבד נתונים על הקשר בין המרצה לבין הקורס. הפתרון הוא לפרק יחס זה לשני היחסים הבאים:

לומד (<u>מספר־סטודנט,</u> מספר־קורס)

מלמד (<u>מספר־מרצה,</u> מספר־קורס).

שני היחסים החדשים מתארים את היחס הקודם בצורה תקנית משופרת באמצעות שני יחסים חדשים המכילים מפתחות בלבד ללא כל תכונה מגדירה.

5.5 צורה תקנית רביעית (4NF—Fourth Normal Form)

יחס הנמצא בצורה תקנית שלישית משופרת עלול עדיין להיות חשוף למספר תופעות לואי לא רצויות. כדי לפתור אותן יש לעבור לצורה התקנית הרביעית.

נשתמש בדוגמא של מפעל שבו מספר מכונות שונות שבכל אחת מהן יכולים להשתמש מספר עובדים ואפשר לייצר מספר מוצרים שונים. כתוצאה מכך קיים מצב שבו לכל מכונה קשורות קבוצה מוגדרת של עובדים וקבוצה מוגדרת של מוצרים שונים. תרשים 8—19 מציג טבלה לא תקנית המתארת מצב זה.

ביחס "ייצור" אין כל תלויות פונקציונליות ולכן המעבר לצורה תקנית שלישית הינו פשוט ומוצג בתרשים 8—20. שורה ראשונה למשל, משמעותה שעובד E1 מייצר מוצר P1 במכונה M1. שאר השורות מציגות עבור כל מכונה את כל הצירופים האפשריים עם שתי התכונות האחרות. היחס החדש נמצא בצורה השלישית המשופרת מכיון שהוא מכיל מפתחות בלבד ללא תכונות מגדירות נוספות. עם זאת יש כאן תופעות לואי מסויימות. למשל, אין אפשרות להוסיף לבסיס הנתונים את הקשר בין מכונה למוצר אלא אם יודעים לפחות עובד אחד המייצר את המוצר על המכונה. בצורה דומה לא ניתן להוסיף את הקשר בין העובד למכונה. קיימות

ייצור

מספר־מכונה	מספר־עובד	מספר־מוצר
M1	E1 E2 E3	P1 P2
M2	E4 E5 E6	P3

תרשים 19 – 8: יחס בצורה לא תקנית

תופעות לואי גם בפעולת ביטול, מאחר וביטול קשר בין מכונה למוצר גורם גם את ביטול הקשר בין עובד למכונה. אם אותו עובד מייצר על אותה מכונה מספר מוצרים שונים ורוצים לבטל את הקשר ביניהם יש לבטל את כל השורות המציגות קשר זה.

ייצור

מספר־מכונה	מספר־עובד	מספר־מוצר
M_1	E_1	P_1
M_1	E_2	P_1
M_1	E_3	P_1
M_1	E_1	P_2
M_1	E_2	P_2
M_1	E_3	P_2
M_2	E_4	R_3
M_2	E_5	P_3
M_2	E_6	P_3

תרשים 20 – 8: יחס תקני

לסילוק תופעות לואי בלתי רצויות אלה יש לפרק את היחס "ייצור" לשני יחסים אשר בהם לא קיימות התופעות שהוזכרו:

מכונה־מוצר (מספר־מכונה, מספר־מוצר)

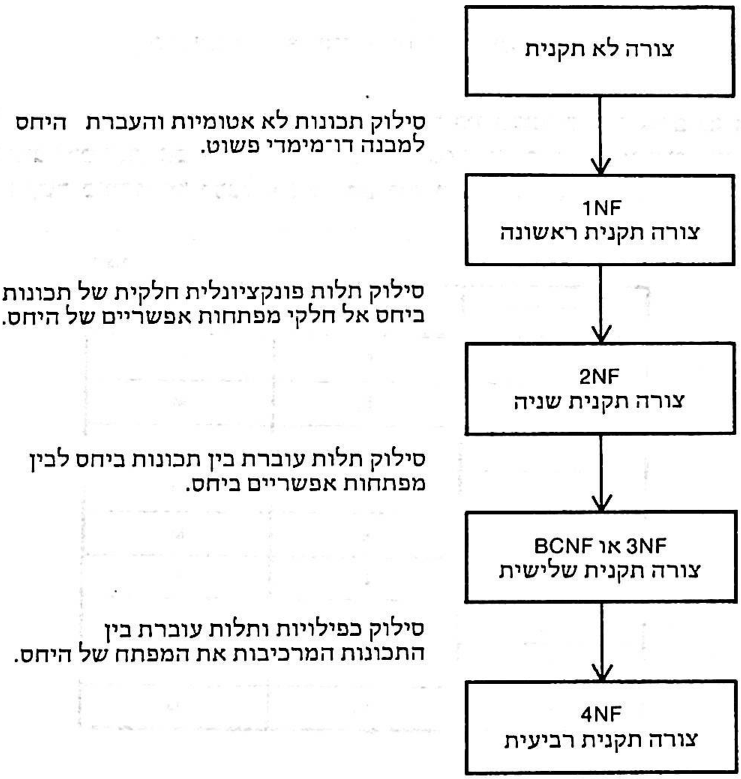
מכונה־עובד (מספר־מכונה, מספר־עובד)

היחס אשר היה בצורה BCNF לא פתר את כל הבעיות ולכן ממשיכים ומפרקים אותו עד לביטול תופעות הלואי השונות. היחס בצורה התקנית השלישית המשופרת מכיל תלות רב־

ערכית (Multivalued Dependency) שהיא הכללה של התלות הפונקציונלית אשר מוגדרת כתלות של מופע בודד במופע אחר. בדוגמא, משמעות התלות הרב־ערכית היא שלמרות שלעובד אין תלות פונקציונלית במכונה עדיין קיימת קבוצה מוגדרת של עובדים שלהם קיימת תלות כזו. כלומר, ישנה תלות של קבוצה של מופעים במופע של תכונה אחרת. המעבר לצורה תקנית רביעית מבטל את התלות הרב־ערכית בין המפתחות של היחס.

5.6 תהליך התיקנון

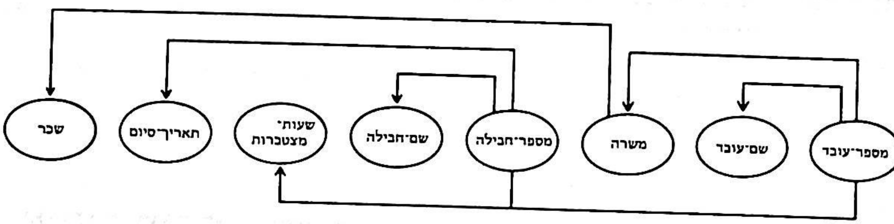
תהליך התיקנון הינו תהליך אלגוריתמי שבעזרתו מועברים יחסים לא מתוקננים ליחסים תואמים אחרים בצורה תקנית רביעית. תרשים 21—8 מסכם את התהליך.



תרשים 21 - 8: תהליך התיקנון

נתאר את התהליך באמצעות דוגמא של היחס "עובד" המציג את התכונות של עובד בחברה המפתחת תוכנה למחשבים. כל עובד עוסק בפיתוח חבילת תוכנה אחת או יותר, ובכל חבילת תוכנה יכולים לעסוק בו־זמנית מספר עובדים. לכל חבילת תוכנה יש תאריך סיום משוער. קיימים ביחס סוגי תלות שונים כפי שמתואר בתרשים 22—8:

עובד (מספר־עובד, שם־עובד, משרה, מספר־חבילה, שם־חבילה, שעות־מצטברות, תאריך־סיום, שכר).



תרשים 22 – 8: התלויות השונות ביחס "עובד"

הקשר הרב—רב־ערכי שבין עובד לבין חבילות תוכנה מחייב מפתח אשר מורכב משתי תכונות: מספר־עובד ומספר־חבילה. ביחס זה קיימים סוגי תלות שונים:

— לשם־העובד ולשכר תלות חלקית במספר־עובד בלבד (חלק מהמפתח),
— לשם־החבילה ולתאריך־הסיום תלות חלקית במספר החבילה,
— לשעות־המצטברות של העובד בפיתוח חבילת התוכנה תלות מליאה במפתח,
— בין השכר לבין מספר העובד יש תלות עוברת דרך המשרה.

היחס מכיל רק ערכים אטומיים ולכן הוא נמצא כבר בצורה תקנית ראשונה. כדי להעבירו לצורה תקנית שניה נסלק את התלויות החלקיות על ידי פירוקו לשלשה יחסים חדשים:

עובד (מספר־עובד, שם־עובד, משרה, שכר)
חבילה (מספר־חבילה, שם־חבילה, תאריך־סיום)
פרויקט (מספר־עובד, מספר־חבילה, שעות־מצטברות)

כדי לעבור לצורה תקנית שלישית יש לסלק את התלות העוברת ביחס "עובד" ונקבל ארבעה יחסים חדשים:

עובד (מספר־עובד, שם־עובד, משרה)
שכר (משרה, שכר)
חבילה (מספר־חבילה, שם־חבילה, תאריך־סיום)
פרויקט (מספר־עובד, מספר־חבילה, שעות מצטברות)

השלב האחרון בתהליך התיקנון יצר מצב שבו אין תלויות בין המפתחות של היחס "פרויקט" (המפתח מורכב משתי תכונות). ארבעת היחסים שנוצרו בתהליך זה מציגים על כן את הצורה התקנית הרביעית, ובכך הסתיים תהליך התיקנון.

תהליך התיקנון מטפל במשמעות של הנתונים ומטרתו הכללית הינה לסלק כפילויות נתונים ולצמצם את הבעיות שעלולות לחול בזמן עדכון בסיס הנתונים. תפקידו להיות בעיקר קו מנחה לקבלת מבנה נתונים טוב יותר ולמניעת תופעות הלואי.

במציאות יש לעיתים להתפשר עם מבנים שאינם מושלמים, בעיקר מתוך שיקולים של יעילות גישה לנתונים. נקח לדוגמא את היחס "עובד" שבו קיימת תלות בין הכתובת למיקוד ולכן הוא אינו נמצא בצורה התקנית הרביעית.

עובד (מספר־עובד, שם־עובד, משרה, כתובת, מיקוד)

הצגת היחס בצורה התקנית הרביעית פירושה פירוקו לשני יחסים שבו אחד מהם מציג את נתוני העובד וכתובתו והיחס השני מציג את הקשר שבין כתובת למיקוד.

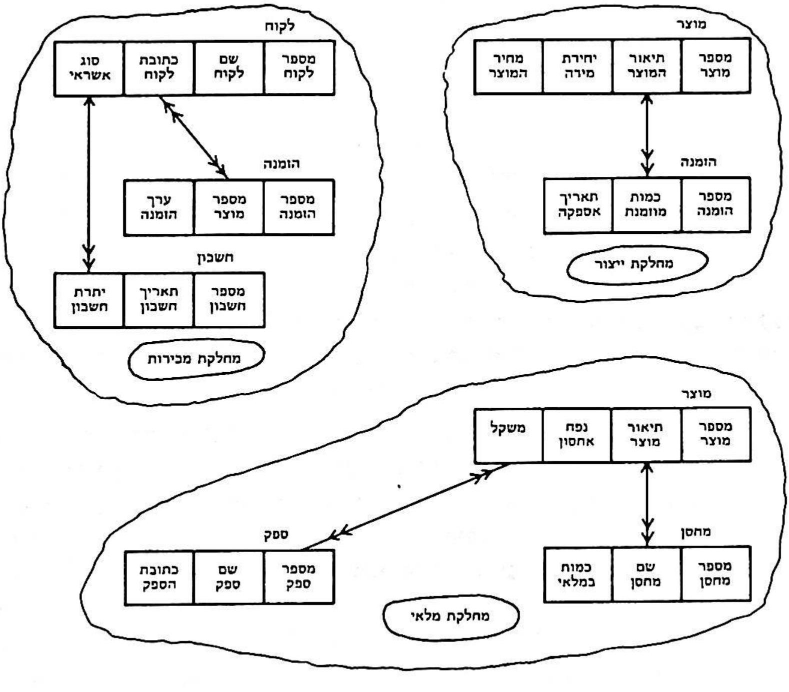
עובד (מספר־עובד, שם־עובד, משרה, כתובת)
מיקוד (כתובת, מיקוד)

אחזור נתונים לשם משלוח הודעה לעובד יחייב פנייה לשתי טבלאות שמשמעותה חוסר יעילות. לכן השיקול הנכון במקרה זה יהיה לא לבחור בצורה התקנית הרביעית.

6. איחוד מודלים של משתמש

מודל הנתונים הינו מרכז וצומת הנתונים של הארגון ולפיכך הוא חייב לשרת מספר רב של משתמשים. לכל משתמש עשוייה להיות תצפית שונה או מודל־משתמש שונה בהתאם לתפקידו בארגון ולסיוע שהוא מבקש לקבל ממודל הנתונים במילוי משימותיו השונות. מודל הנתונים של הארגון הוא אחד בעוד שכל משתמש מרכיב לעצמו מודל משתמש מבלי להתחשב בצרכיהם של אחרים. תפקידו של מנהל בסיס הנתונים **לאחד את המודלים של המשתמשים** (View Integration) וליצור מודל משותף אשר יוכל לשרת את הכל.

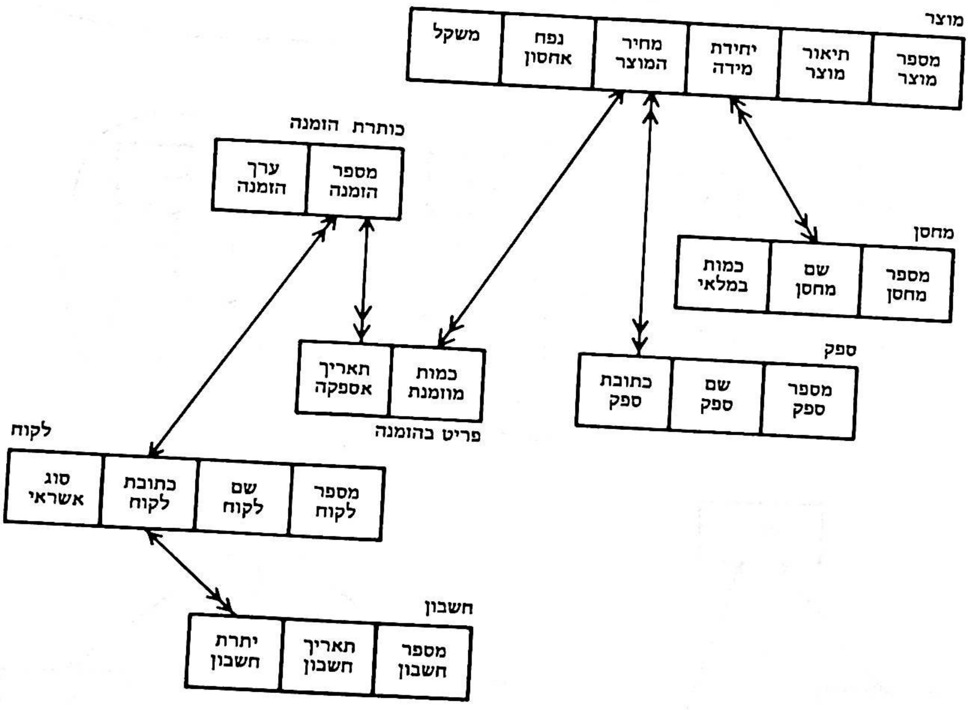
בשלב ראשון על המתכנן לזהות את כל הישויות, התכונות והקשרים עבור כל אחד מהמשתמשים במערכת. עליו לבחון מקרים של חוסר עקביות במבנה הנתונים, בקביעת השמות שלהם או אף בתוכנם. בתרשים 23—8 מוצגים לדוגמא שלשה מודלים שונים של נתונים במפעל ייצור: המודל של מחלקת הייצור, המודל של מחלקת המכירות והמודל של מחלקת המלאי. כל מחלקה עוסקת בתחום אחר ולכן הנתונים שהיא משתמשת בהם שונים והם משתלבים בצורה שונה. למרות זאת נוכל להבחין שישנם גם נתונים זהים.



תרשים 23 – 8: שלשה מודלים חיצוניים במפעל ייצור

במודלים אלו יש חוסר עקביות המתבטאת בכך שישנן ישויות בעלות שמות זהים אשר כוללות תכונות שונות. לדוגמא לישות "מוצר" במודל מחלקת הייצור יש תכונות שונות מאלו של הישות "מוצר" במודל מחלקת המלאי. יתכנו גם מצבים שלתכונות בעלות שמות זהים תהיינה משמעויות שונות במודלים שונים. מצב של חוסר עקביות בתיאור הקשרים בין הישויות קיים גם־כן. במודל מחלקת הייצור למשל, כל הזמנה מתייחסת למוצר אחד בשעה שמודל מחלקת המכירות מניח שכל הזמנה כוללת מספר מוצרים שונים. חלק מחוסר העקביות ניתן לפתרון על ידי שימוש במילון נתונים אשר מתאר את כל הישויות והתכונות וקובע סטנדרטים לשימוש בהן.

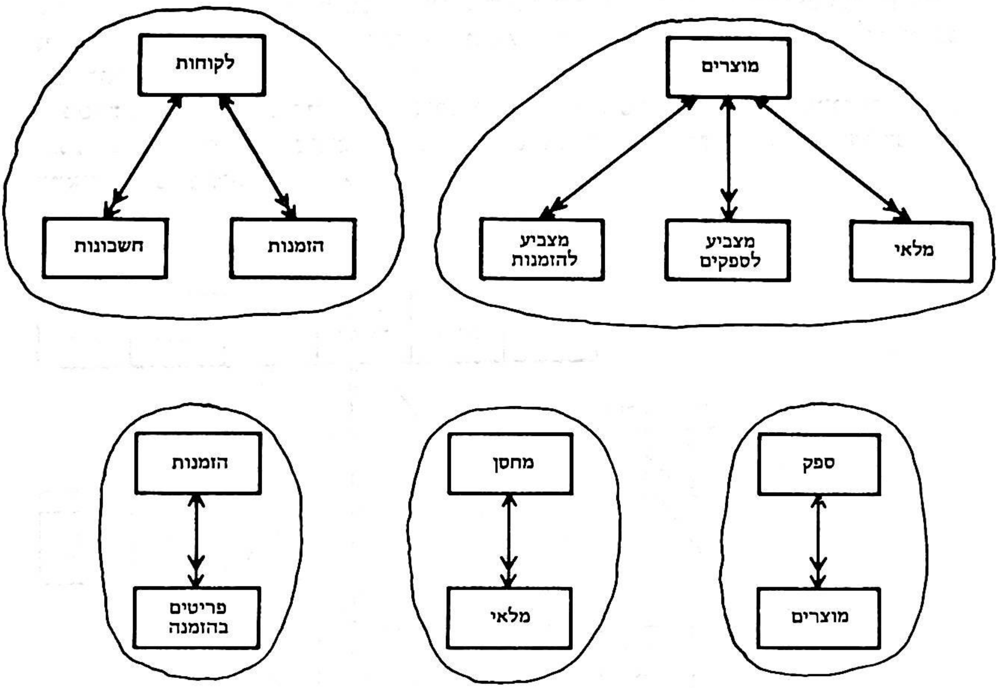
מטרת איחוד המודלים הינה ליצור מודל אחד לארגון, אשר יכיל את כל הישויות והתכונות, וממנו יגזרו על ידי המשתמשים השונים המודלים אשר דרושים להם בעבודתם. מודל הנתונים המאוחד מוצג בתרשים 24—8.



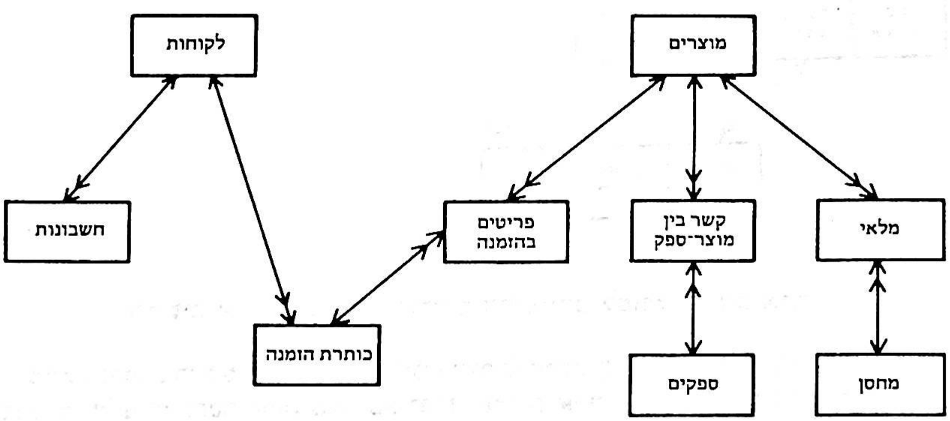
תרשים 24 – 8: איחוד המודלים החיצוניים למודל נתונים אחד

מודל הנתונים הינו מודל תפישתי שאינו קשור למערכת DBMS כלשהי מכיון שאינו עוסק בבעיות של צורת אחסון הנתונים ושיטת הגישה אליהם. אי התלות של המודל במערכת DBMS כלשהי הינו חשוב מפני שהוא מאפשר את יישומו בכל מערכת שתיבחר, במערכת שהארגון מפעיל או במהדורות חדשות של מערכת קיימת. מודל הנתונים התפישתי הינו חשוב מפני שהוא מתאר את המציאות בצורה הטובה ביותר האפשרית ואינו מכיל מלכתחילה מגבלות

הנובעות משימוש במערכת מסויימת. מודל הנתונים שתוכנן כראוי ישאר בתוקף לאורך זמן ללא שינויים או עם שינויים מעטים, וכאשר יהיה צורך לשנותו תהייה בו לרוב גמישות מספקת לשילוב השינוי ללא קושי.



א. סכימת מודל נתונים היררכי



ב. סכימת מודל נתונים רשתי

תרשים 25 – 8: המרת המודל התפישתי (24 – 8) למודלים לוגיים

מודל הנתונים התפישתי אינו בר־יישום בצורה ישירה. יש להמיר אותו לסכימה המתאימה למערכת DBMS שנבחרה על־ידי העברת הישויות והקשרים למבנים שהיא משתמשת בהם. במערכת המבוססת על המלצות CODASYL תכלול ההמרה העברה למבנים של סוגי־רשומה וחבורות ושימוש בקשרים חד—רב־ערכיים בלבד. תרשים 25—8 מציג מבנים אפשריים של סכימה הירכית וסכימה רשתית המבוססות על מודל הנתונים שבתרשים 24—8. בשתי הסכימות מופיעים מבנים מיוחדים נוספים הדרושים לתרגום המודל הכללי למודל הירכי או למודל רשתי. בעת בניית הסכימות למודלים על המתכנן לפעול בהתאם לכללי התכנון ולשקול יעילות ומהירות תגובה ולקבוע מסלולי גישה לרשומות בהתאם לתדירות השימוש בהן.

7. שאלות

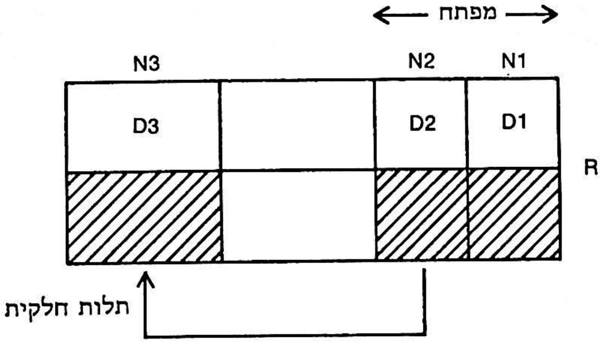
1. מהו ההבדל בין תלות פונקציונלית לתלות פונקציונלית מלאה?
2. בנה מודל אופטימלי עבור המקרה הבא:

מרצה מלמד במספר קורסים. קורס מסויים מועבר ע"י מספר מרצים שונים. לכל קורס יתכנו מספר ספרי לימוד. ספר לימוד יכול לשמש יותר מקורס אחד. כל המרצים המלמדים את אותו הקורס משתמשים באותו הספר.

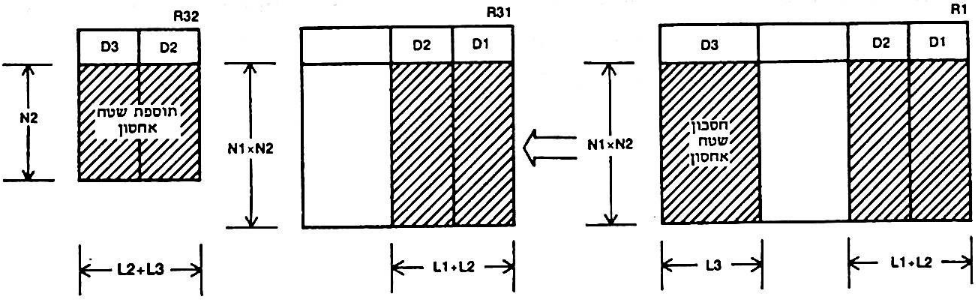
3. האם מודל נתונים חיצוני המשמש אך ורק לאיחזור נתונים חייב להיות מתוקנן במלואו?
4. התייחס למערכת רגילה לעיבוד נתונים המשתמשת בקבצים ורשומות. האם רעיון התיקנון ישים גם בבניית קבצים אלה ואם כן, באיזו צורה.

8. נספח – שיקולי שטח אחסנה במעבר לצורה תקנית שלישית

אחסון יחסים בצורתם התקנית השלישית עשוי להיות חסכוני יותר מאשר אחסון בצורה התקנית הראשונה או השנייה. נציג זאת על ידי תיאור המעבר מצורות תקניות אלו והחסכון בשטח אחסון הנובע מכך.



תרשים 26 – 8: יחס בצורה תקנית ראשונה



תרשים 27 – 8: העברת היחס R_1 לצורה תקנית שלישית

8.1 מעבר מצורה תקנית ראשונה

נבדוק תחילה את השפעת סילוק התלות החלקית על נפח האחסנה הדרוש. נעזר בתרשים 26—8 שבו נמצא יחס R1 בצורה התקנית הראשונה. המפתח מורכב משתי תכונות בעלות תחום ערכים D1 ו־D2 אשר יכולים להכיל N1 ו־N2 ערכים שונים בהתאמה. לתכונה D3 אשר יכולה להכיל N3 ערכים שונים תלות חלקית ב־D2.

בעת מעבר מהצורה התקנית הראשונה לצורה התקנית השלישית יוצרים שני יחסים חדשים R31 ו־R32 אשר יש להם נפח אחסנה שונה מאשר נפח האחסנה של יחס R1. נחשב את ההפרש בשטח האחסנה ונניח שמספר התווים הדרושים לאחסון כל תכונה הוא L1, L2, L3 בהתאמה.

אם ל־D1 יש N1 ערכים שונים ול־D2 יש N2 ערכים שונים, אזי לכל היותר יתכנו N1×N2 ערכים שונים ביחס R1. זה יהיה המספר המירבי של שורות ביחס R1 וכן גם ביחס R31 אשר נגזר ממנו עם התכונות D1, D2. **החסכון** בשטח האחסנה יהיה על־כן N1×N2×L3 **והתוספת** בשטח האחסנה תהיה N2(L2+L3). כדי שהחסכון יהיה גדול מהתוספת חייב להתקיים:

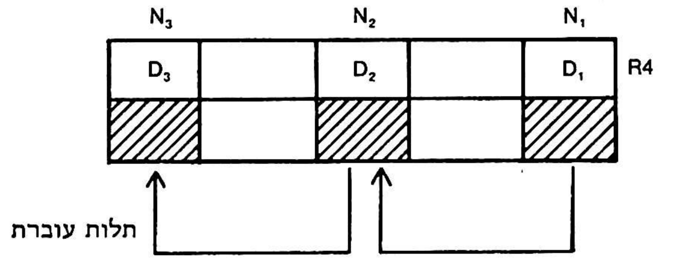
$$N2(L2+L3) < N1 \times N2 \times L3$$

$$N1 > \frac{L2 + L3}{L3}$$

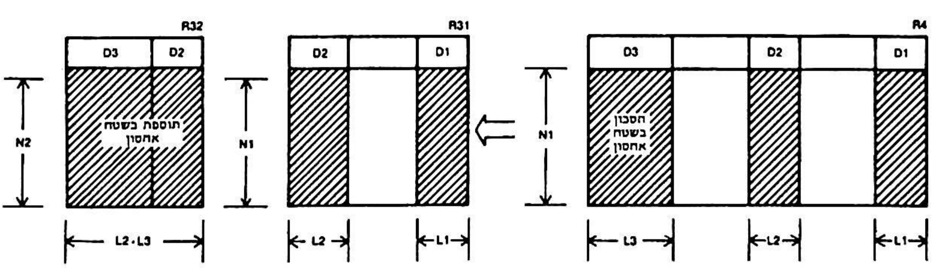
אם מתקיים $L2 \cong L3$, התנאי מתקיים עבור $N1=2$, כלומר כמעט תמיד ניתן לחסוך שטח אחסנה על ידי מעבר מצורה תקנית ראשונה לצורה תקנית שלישית. אם L2 גדול מאד באופן יחסי ל־L3 ואם N2 קטן מאד תדרש תוספת של שטח אחסנה.

8.2 מעבר מצורה תקנית שנייה

היחס R4 בתרשים 8—28 מוצג בצורה תקנית שניה ויש בו תלות עוברת בין תכונה D1 ל־D3 באמצעות התכונה D2. לכל תכונה תחום ערכים שבו N1, N2, N3 בהתאמה.



תרשים 28 - 8: יחס בצורה תקנית שניה



תרשים 29 - 8: העברת היחס R4 לצורה תקנית שלישית

התנאי לחסכון בשטח אחסנה במקרה זה יהיה:

$$N1 \times L3 > N2(L2+L3)$$

$$\frac{N1}{N2} > \frac{L2+L3}{L2}$$

אם $L3 \cong L3$ נקבל כשהיחס בין N1 ו־N2 צריך להיות גדול מ־2 אם N1 למשל מייצג את מספר העובדים במפעל ו־N2 את מספר המשרות במפעל נובע שאם לכל משרה יש יותר משני עובדים נקבל חסכון בשטח האחסנה. התנאי לחסכון מצביע על כך שהמעבר מהצורה התקנית השניה לצורה התקנית השלישית גורר גם כן חסכון באחסנה.

לסיכום אפשר לומר שהצורה התקנית השלישית הינה חסכונית בשטח אחסנה כאשר מתקיימים התנאים לכך. בנוסף יש לצורה תקנית זו יתרונות אחרים שכבר עמדנו עליהם.

פרק ט

שלימות ובטיחות נתונים

(Data Integrity and Security)

1. מבוא

בפרק זה נעסוק בבעיות של נכונות, איכות וזמינות של הנתונים המאוחסנים בבסיס הנתונים. אפשר לכלול זאת במונח **אמינות** אשר מציין שבסיס הנתונים מייצג בכל עת בצורה נכונה את המציאות התפעולית של הארגון. האמינות של בסיס הנתונים מושגת בשלש דרכים עיקריות אשר נדון בהן בהמשך:

א. **שלימות** (Integrity) היא האפשרות של מערכת DBMS להגן על הנתונים מפני תקלות תוכנה וחומרה. עליה לדאוג לכך שהנתונים המאוחסנים בבסיס הנתונים יקבלו את הערכים שהמשתמש קבע, ולמנוע רישום ערכים שגויים הנובעים מבעיות תוכנה, או מהסביבה התפעולית של המערכת.

ב. **התאוששות** (Recovery) היא האפשרות של מערכת DBMS להתאושש מתקלות כגון תקלות בתוכניות היישום, הפסקת חשמל, תקלות חומרה. היכולת להתאושש מתקלות היא קריטית מכיון ששיתוק בסיס הנתונים יכול להביא לשיתוק פעולות הארגון כולו. כבכל מערכת תפעולית כך גם במערכות מידע ממוחשבות לא ניתן להמנע מתקלות שונות אשר עלולות לפגוע באמינות בסיס הנתונים. ריכוז כל הנתונים במאגר אחד ותלות פעילות הארגון במאגר זה מחייבים לספק פתרונות הולמים להתאוששות מתקלות אלה.

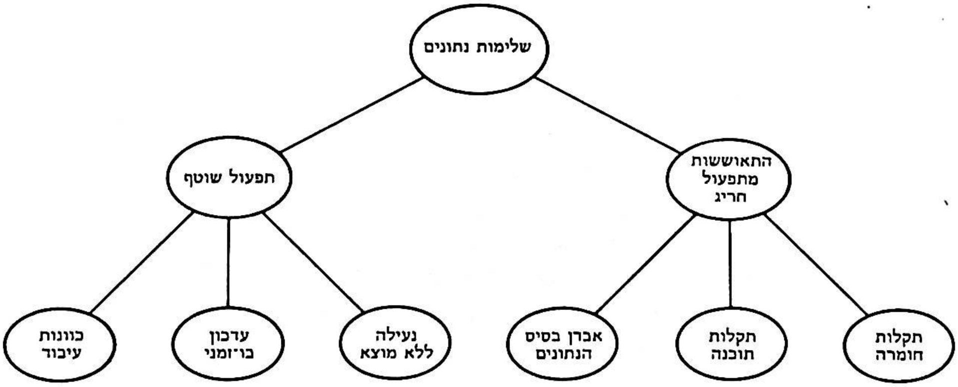
ג. **בטיחות** (Security) היא האפשרות של מערכת DBMS להגן על הנתונים מפני גישות בלתי רצויות בשוגג או במזיד לשם קריאת הנתונים או לשם עדכונם.

2. שלימות נתונים

במערכות עיבוד נתונים המבוססות על קבצים נמצא כל קובץ בבעלות נפרדת. בעל הקובץ אחראי לעדכונו והוא אשר קובע את מהלך תוכנית הקליטה אשר מזינה את הנתונים למערכת. בעל הקובץ יכול להתיר גישה לקובץ לשם קריאה ממנו על ידי אחרים אך הוא לא יתיר להם לעדכן אותו. תהליך העדכון של הקובץ הבודד אינו עוסק בדרך כלל ביחסים של הקובץ עם קבצים אחרים והוא רואה בו יחידה עצמאית.

במערכות בסיסי נתונים המצב שונה. בסיס הנתונים נמצא באחריות ובבעלות הארגון המיוצג על ידי מנהל בסיס הנתונים והגישה אליו מותרת לכל בכפיפות לנוהלי הסודיות הנקבעים בארגון. העדכון של החלקים השונים בבסיס הנתונים הינו באחריות בעלי הנתונים. נתוני כח אדם יעודכנו למשל על ידי מחלקת השכר ונתוני תמחיר — על ידי מחלקת התמחיר. מידת חלוקת האחריות לנתונים שונה בכל מקרה ולא נדון בה כאן.

העדכון של בסיס נתונים מורכב יותר מאשר העדכון בקבצים נפרדים מכיון שקיימת תלות בין הנתונים השונים אף־על־פי שהם נמצאים בבעלויות נפרדות. קושי נוסף יכול להיות בכך שבתוך יחידת נְתונים אחת יכולים חלקים שונים להתעדכן על ידי גופים שונים בארגון. לדוגמא, רשומת עובד נמצאת באחריות מחלקת כח־אדם. מחלקה זו תעדכן את פרטי הזיהוי שלו ותהייה אחראית לשלימות הישות הקרויה "רשומת־עובד", אך מחלקת ההדרכה תורשה לעדכן את נתוני ההשתתפות בקורסים ומחלקת השכר תעדכן את הנתונים שבתחום טיפולה. העדכון של ישות אחת "רשומת־עובד" עשוי להיות בו־זמני על ידי כל אחת מהמחלקות שקבלו הרשאה לכך. המערכת חייבת לסַנכרן את העדכון ולדאוג לכך שהתוצאה הסופית תשקף את המציאות שהמעדכנים רוצים להציג. הישות "רשומת־עובד" קשורה לישות אחרות במערכת ועל־כן הוספה או ביטול של ישויות עובד מחייב תאום בתחום בסיס הנתונים. הרחבנו את הדיון בפעולת העדכון מכיון שהיא עלולה להיות אחד הגורמים לפגיעה בנתונים.



תרשים 1 – 9: שלימות הנתונים – תקלות אפשריות

מערכת בסיס נתונים משרתת יישומים שונים הפועלים בו־זמנית, ובדרך כלל במערכת מקוונת. יש תלות גבוהה של המשתמשים בפעילות התקינה והרצופה של המערכת ובזמינות בסיס נתונים אמין. ריבוי המשתמשים יוצר איום גדול לאמינות בסיס הנתונים ומקשה על תהליכי גילוי התקלות. פעולת התיקון של תקלה שמתגלית הינה מורכבת וקשה בגלל שילוב הנתונים במערכת ובגלל ההכרח לבצע את התיקונים תוך כדי פעולה רצופה שלה.

נחלק את בעית השלימות לשתים: בעיות הנובעות **מתפעול שוטף ורגיל** (Normal Mode) ובעיות הנובעות **מתפעול חריג** (Abnormal Mode). הבעיות העיקריות של מערכת DBMS בתפעול שוטף נובעות בעיקר מהאפשרות שמספר תוכניות יישום עובדות בו־זמנית מול אותם נתונים ומעדכנות אותם או חלק מהם. הבעיות העיקריות בתפעול חריג נובעות מתקלות שונות של תוכנה או חומרה ומדרכי ההתאוששות מתקלות אלה.

2.1 מקורות השגיאות בבסיס הנתונים

פגיעה בשלימות בסיס הנתונים יכולה לנבוע משגיאות ותקלות ממקורות שונים. נסקור בקצרה את המקורות האפשריים לשגיאות:

א. **שגיאות משתמש:** המשתמשים השונים בבסיס הנתונים (במסופים למשל) או תוכניות היישום המעדכנות את בסיס הנתונים הם מקור אפשרי לפגיעה בשלימות. דוגמא אפשרית היא הזנת נתונים לא נכונים לתכונות, כמו למשל קביעת דרגה מקצועית לא נכונה לעובד ברשומת כוח אדם, תקלה אשר ההגנה מפניה כמעט אינה אפשרית. דוגמא אחרת היא תוכנית יישום כלשהי אשר מעדכנת את השדה הלא נכון ברשומה. המנגנונים השונים המטפלים בשלימות בסיס הנתונים אינם יכולים למנוע הופעת נתונים שגויים אם המשתמש הכניס אותם בכוונה תחילה. מנגנונים אלה מטפלים בעיקר בשגיאות אשר עלולות לנבוע מתוכנת מערכת DBMS או ממערכת ההפעלה.

ב. **שגיאות תוכנה:** תקלות בתוכנת מערכת DBMS עלולות לגרום לפגיעה בשלימות בסיס הנתונים, כמו למשל כאשר חל **סיום חריג** (Abnormal Termination) במהלך פעולת העדכון. סיום חריג זה עלול להשאיר את בסיסי הנתונים במצב לא ברור מכיון שחלק מהרשומות עודכנו בעוד שחלק מהן לא עודכנו עדיין. סיבה אחרת לפגיעה בשלימות בסיס הנתונים יכולה להיות השיטה שבה מערכת DBMS מטפלת בעדכון בו זמני של בסיס הנתונים על ידי מספר תוכניות יישום שונות.

ג. **שגיאות ותקלות חומרה:** שגיאות ותקלות חומרה הן בדרך כלל בעלות הסתברות הופעה נמוכה יחסית אך הן חמורות מאד עם הופעתן. לדוגמא תקלה במשטח מגנוט בדיסק מגנטי אינה רצויה כמובן אך היא חמורה יותר אם משטח זה מכיל מצביעים או אינדקסים, כי אין אפשרות להמשיך לתפעל את בסיס הנתונים.

ד. **תקלות סביבתיות:** בין הנפוצות בתקלות אלו הן נפילת מתח חשמלי רגעית, הפסקת חשמל או הפסקת מערכת מיזוג האויר אשר גורמות להפסקת פעילות. הפסקת פעילות פתאומית יכולה לגרום לשיבושים ולפגיעה בבסיס הנתונים. מפני תקלות סביבתיות ניתן להגן באמצעות מערכות גיבוי מתאימות אשר לא נדון בהן כאן.

3. טכניקות לשמירה על שלימות הנתונים

מערכות DBMS מספקות בדרך כלל מספר מנגנונים כדי להגן על שלימות בסיס הנתונים תוך כדי תפעול שוטף. נסקור מספר טכניקות המכוונות למטרה זו.

3.1 כוונת עיבוד (Processing Intent)

מנגנון זה מטפל בהרשאה לעיבוד בסיס הנתונים על ידי תוכנית היישום ובהשלכות על תוכניות אחרות במערכת. לפני שתוכנית היישום מתחילה לעבד את בסיס הנתונים עליה להצהיר מה בכוונתה לעשות: לקרוא בלבד או לקרוא ולעדכן. הצהרה זו מבוצעת בדרך כלל בפקודת OPEN של בסיס הנתונים ועל פיה ניתנת ההרשאה. ברוב המערכות מוכרות שלש צורות של הרשאה:

— **בלבדי** (Exclusive): אם תוכנית היישום מצהירה שבכוונתה לעבד את בסיס הנתונים באופן בלבדי, לא תנתן הרשאה לכל תוכנית אחרת לגשת לבסיס הנתונים באותו זמן. לדוגמא, תוכנית המחשבת את שכר העובדים במפעל לא תרצה שתוכנית יישום אחרת תעדכן נתונים ברשומות, תבטל או תוסיף רשומות. הגישה של תוכניות אחרות יכולה להיות אסורה בכלל או שהיא יכולה להיות מוגבלת לחלקי בסיס הנתונים או לרשומות שאינן בטיפול תוכנית היישום שבה אנו דנים. לדוגמא, אפשר למנוע גישה אל רשומת המלאי כאשר מעדכנים אותה אך לאחר מכן רשומה זו "תשוחרר" לשימוש והגישה תמנע לרשומה אחרת שבתהליך עדכון.

— **מוגן** (Protected): כאשר תוכנית היישום מצהירה שבכוונתה לעבד את בסיס הנתונים באופן מוגן, תורשה הגישה לתוכניות יישום אחרות למטרת קריאה בלבד ולא תורשה הגישה לתוכניות אלו לשם עדכון.

— **משותף** (Shared): הצהרה של עיבוד משותף של בסיס הנתונים מתירה גישה של כל סוגי התוכניות לצורך קריאה ולצורך עדכון. גישה לקריאה משותפת מותרת תמיד (אלא אם יש הגבלת סודיות שעליה נעמוד בהמשך) אך גישה משותפת לעדכון יכולה להיות רק כאשר מובטח מעל לכל ספק שלא יווצר מצב של עדכון בו־זמני של נתונים על־ידי תוכניות שונות.

3.2 עדכון בו־זמני (Concurrent Update)

הנחנו עד עתה שאם תוכניות היישום פועלות כשורה בסיס הנתונים יכיל רק נתונים אמינים ונכונים. מסתבר שהנחה זו אינה מדוייקת מכיון שיתכנו מצבים שבהם תוכניות היישום פועלות כשורה ובכל זאת תוכן בסיס הנתונים אינו אמין. בעייה זו נובעת מהאפשרות של עבודה במקביל של מספר תוכניות יישום על אותם הנתונים.

אחד היתרונות הגדולים של מערכת מקוונת הינו זמן התגובה המהיר אשר מושג על־ידי האפשרות של מספר תוכניות שונות לפעול במקביל ובאופן **בו־זמני** (Concurrently) על בסיס נתונים אחד. מערכת DBMS ערוכה לפעילות בו־זמנית והיא מפקחת על כך. גישה של תוכניות לבסיס נתונים לפי תור היתה גורמת זמן תגובה ארוך, וגישה ללא פיקוח היתה גורמת לפגיעה בנתונים. במערכת מקוונת של הזמנת כרטיסי טיסה למשל נדרש זמן תגובה קצר אך נדרשת גם הגנה על הנתונים מפני עדכון לא מסודר של נתוני מכירת מושבים בטיסה אחת על ידי סוכני נסיעות שונים.

בסיס נתונים יוגדר **אמין** כאשר הוא מכיל נתונים שלימים ונכונים בלבד. אך במהלך פעולת

תוכנית היישום, אמינות בסיס הנתונים מופרת באופן זמני. במערכת מידע רפואית למשל הכנסת חולה חדש למחלקה צריכה להגדיל את מונה מספר המיטות התפוסות שברשומת המחלקה. ברגע נתון כלשהו יתכן שתוכנית היישום כבר הספיקה לכתוב את רשומת החולה החדש בבסיס הנתונים אבל טרם הספיקה לשנות את המונה. בסיס הנתונים אינו אמין בנקודת זמן זו, ולכן אם מסיבה כלשהי תופסק ברגע זה עבודת המערכת הוא יכיל נתונים לא נכונים. רמה נמוכה של כפילות נתונים עשוייה למנוע תקלה כזו אך לא ניתן למנוע אותה כליל. אחת השיטות לטפל בבעייה הינה רישום שוטף של כל התנועות המוזנות למערכת ו"שחזור" העדכון לאחר שחלה תקלה במהלך הביצוע שלו.

בדרך כלל נהיה מעוניינים להפעיל כמה שיותר תוכניות יישום במקביל בתנאי שהן רק קוראות את הנתונים. אפשרות הפעלת מספר רב של תוכניות במקביל תשפר את זמן התגובה של כל תוכנית, אך מאידך יש למנוע הפעלה במקביל של תוכניות המעדכנות את אותו הנתון. שיתוף הנתונים בין התוכניות קיים כל עוד אין עדכון משותף של נתונים.

העדכון הוא פעולה בסיסית וחשובה במערכת ועל כן מן הראוי להציג איך יכול עדכון בו־זמני של נתון כלשהו להפר את אמינות בסיס הנתונים. נניח שמשני מסופים מנסים לעדכן את יתרת המלאי של פריט אשר לפני העדכון היתה 10 יחידות. ממסוף A רוצים להוסיף למלאי 5 יחידות וממסוף B רוצים להקטין את המלאי ב־2 יחידות. התוצאה הסופית הנכונה צריכה להיות 13 יחידות במלאי. עתה נראה כיצד סדר אירועים לא מבוקר עלול להפיק תוצאה שונה. ראה תרשים 2—9.

סדר האירועים	תוכן בסיס הנתונים	הוראות תוכנית מסוף A	הוראות תוכנית מסוף B
1	10		
2	10	קרא רשומת הפריט לזכרון והעבר לשטח העבודה.	
3	10		קרא רשומת הפריט לזכרון והעבר לשטח העבודה
4	10	הוסף 5 יחידות לשדה "יתרת מלאי"	
5	15	כתוב רשומה מעודכנת בבסיס הנתונים	
6	15		החסר 2 יחידות משדה "יתרת מלאי"
7	8		כתוב רשומה מעודכנת בבסיס הנתונים

תרשים 2 – 9: שתי תוכניות במקביל ללא מנגנון נעילה

אם סדר האירועים מודגם בתרשים 2—9 התוצאה הסופית תהיה 8 והיא אינה התוצאה הנכונה. הטעות נובעת מהעובדה ששתי התוכניות הפועלות בשירות מסופים A ו־B בהתאמה קראו לזכרון את רשומות המלאי המקורית, עוד לפני שהתוכנית האחרת הספיקה לעדכן אותה. כתוצאה מכך פעלו שתיהן על יתרת מלאי של 10 יחידות וכתבו את התוצאות הנובעות מכך לבסיס הנתונים. אם תוכנית מסוף B היתה כותבת תחילה ולאחריה התוכנית של מסוף A, היתה התוצאה הסופית 15 ולא 8. כמובן שגם תוצאה זו אינה נכונה.

הפתרון לבעייה זו הוא הפעלת **מנגנון נעילה** כלשהו כדי שהתוכנית הראשונה המבקשת את הנתון תיהפך באופן זמני **לבעלים** (Owner) של הנתון. כל שאר התוכניות המבקשות לגשת לאותו נתון תצטרכנה להמתין עד שהתוכנית הראשונה תשחרר אותו. נניח שקיימת הוראה מיוחדת LOCK המאפשרת נעילת נתון בבסיס הנתונים. בתוכנית היישום היא תרשם לפני פקודות הקריאה והעדכון. לאחר כתיבת הרשומה לבסיס נתונים תרשם ההוראה המקבילה לה UNLOCK אשר תשחרר את הנתון לשימושן של תוכניות אחרות. כלומר, ניתן להגן על בסיס הנתונים על ידי נעילה חלקית שלו ופעולה בלבדית של התוכנית על רשומה או נתון ולא על בסיס הנתונים כולו. לאפשרות זו השפעה רבה על שיפור זמן התגובה.

סדר האירועים	תוכן בסיס הנתונים	הוראות תוכנית מסוף A	הוראות תוכנית מסוף B
1	10		
2		נעל את רשומת הפריט	
3	10	קרא רשומת הפריט לזכרון והעבר לשטח העבודה	
4	10		נעל את רשומת הפריט (מכיון שהפריט כבר נעול, תוכנית זו נכנסת למצב של המתנה)
5	10	הוסף 5 ליתרת המלאי	
6	15	כתוב רשומה מעודכנת בבסיס הנתונים	
7	15	שחרר את הרשומה	
8	15		קרא רשומת הפריט לזכרון והעבר לשטח העבודה
9	15		הורד 2 מיתרת המלאי
10	13		כתוב רשומה מעודכנת בבסיס הנתונים
11	13		שחרר את הרשומה

נעילת רשומה

שחרור רשומה

תרשים 3 – 9: שתי תוכניות במקביל עם מנגנון נעילה

3.3 מנגנון הנעילה (Lock Function)

מנגנון הנעילה פועל כמנגנון סינכורוניזציה בין תוכניות כמתואר בתרשים 3—9. כאשר תוכנית מבקשת לנעול נתון כלשהו היא תורשה לעשות זאת ולהמשיך בפעולתה. אם הנתון כבר נעול היא תכנס לתור התוכניות הממתינות לו עד אשר התוכנית הקודמת לה תשחרר את הנעילה.

מנגנון הנעילה גורם מספר בעיות אשר בהמשך נעמוד על החשובות שבהן.

3.3.1 יעד הנעילה

יעד הנעילה הוא תחום בסיס הנתונים אשר כפוף להוראת הנעילה. טבעי הוא לנעול תחום קטן ככל האפשר כדי להתיר פעילות מקבילית בו־זמנית רבה ככל האפשר. עם זאת נעילה ברמה נמוכה יותר (שדה לעומת קובץ למשל) גורמת למורכבות רבה יותר של המערכת וכתוצאה גם העלות גבוהה יותר. נתאר את האפשרויות ביעדי נעילה שונים.

שדה: יעד הנעילה יכול להיות שדה בודד בתוך רשומה. בדוגמא הקודמת מעדכנת תוכנית היישום רק את שדה יתרת־מלאי ולכן נמנעת מתוכניות אחרות הגישה אליו בלבד. אין סיבה למנוע עדכון שדות אחרים באותה רשומה על ידי תוכניות אחרות. נעילת שדה בודד הינה מורכבת מאד ועל כן מעטות מערכות DBMS המציעות אותה. מנגנון בקרה לא אמין יכול לגרום את הבעיות שהוצגו בסעיף עדכון בו־זמני ובתרשים 2—9.

רשומה: מכיון שכל פקודות הקלט/פלט בתוכנית מתייחסות לרשומה, ניתן לקבוע אותה כיעד נעילה. בשיטה זו נמנע מתוכניות יישום אחרות לגשת לרשומה הנעולה אך תהייה להן גישה לרשומות אחרות.

גוש: פעולות קלט/פלט פיסיות מתבצעות בגושים ולא ברשומות בודדות ועל־כן הגוש הינו יעד נעילה אפשרי נוסף. אם רשומה אחת עשוייה להתפרס על פני מספר גושים מנגנון הנעילה צריך לאפשר את נעילת כולם. בשיטה זו תכנסנה להמתנה תוכניות יישום אשר פונות לרשומה אחרת מזו המעודכנת אבל נמצאת באותו גוש פיסי. שיטה זו נוחה יותר ליישום אך מקטינה את אפשרויות העבודה במקביל.

קובץ: קיימות מערכות המבצעות נעילה של כל הקובץ בזמן עדכון רשומה כלשהי. שיטה זו פשוטה ליישום אבל מטילה הגבלות חמורות אפילו על עדכון רשומות שונות על ידי תוכניות נפרדות.

3.3.2 משך הנעילה

משך הנעילה מוגדר כזמן העובר מרגע הנעילה ועד השחרור. רצוי שמשך זמן זה יהיה קצר ביותר כדי לאפשר מירב פעילות בו־זמנית וזמן תגובה קצר. מומלץ לקבוע את פקודות הנעילה רק באותו קטע שבו התוכנית עומדת לבצע את העדכון ולא לפני כן, כדי שמידת הפגיעה בתוכניות אחרות יהיה קטן ככל האפשר.

כדי להקטין את משך הנעילה קיימות מערכות בהן פקודת הנעילה נועלת את כל הגישות ליחידת האחסון ולכן ראש הקריאה/כתיבה נשאר מוצב במשך כל זמן העיבוד מעל המסילה שבה נמצאת הרשומה. כאשר מסתיים העיבוד והתוכנית כותבת את הרשומה נעשה הדבר במהירות יחסית ללא צורך בהזזת מנגנון הזרועות, ויש בכך יתרון ניכר. קיימות מערכות הפעלה אשר משתמשות בשיטת נעילת ראש הקריאה/כתיבה של יחידת האחסון בעת ביצוע

פעולות מסויימות בספריות המערכת.

3.3.3 נעילה ללא מוצא (Deadlock)

השימוש במנגנון הנעילה עלול להביא למצבים שבהם שתי תוכניות תכנסנה למצב של נעילה ללא מוצא מכיון שכל אחת מהן תנעל משאב שהשניה מבקשת להשתמש בו. ישנן למשל שתי תוכניות, האחת נועלת קודם את רשומה A ואחר כך את B, ואילו התוכנית השניה נועלת קודם את רשומה B ואחר כך את רשומה A. הפעלה מקבילה של שתי התוכניות עלולה לגרום למצב של נעילה ללא מוצא כי התוכנית הראשונה נעלה את A וממתינה ל־B בעוד שהתוכנית השנייה נעלה את B וממתינה ל־A.

למניעת מצבי נעילה ללא מוצא הוקדשו מאמצי מחקר רבים בעיקר בגלל חשיבות הנושא בתכנון מערכות הפעלה למחשבים. קיימות מספר שיטות למניעת מצבים אלה וכדוגמא נציג שתים מהן:

— כל תוכנית צריכה לבצע את כל הנעילות שלה בבת־אחת. עד לגמר ביצוע הנעילות מערכת ההפעלה לא תאפשר לאף תוכנית אחרת להתחיל לעבוד.
— סדר הנעילות הוא קבוע ויתקיים בסדר זה בכל תוכניות היישום. לדוגמא, סדר הנעילות יהיה A, B, C, M, X.

כאשר יש מצב של נעילה ללא מוצא יכולה מערכת ההפעלה ל"פתור" אותו. היא תבצע סקירה תקופתית של התוכניות המתבצעות ובעזרת אלגוריתם הבנוי בה היא תאתר מצבים של נעילה ללא מוצא. כאשר היא תגלה מצב כזה היא תגרום לאחת התוכניות להפסיק פעולתה ולהתחיל מחדש. במקרה זה יש לבטל תחילה את כל העדכונים שהתוכנית שהופסקה ביצעה בבסיס הנתונים. פעולה זו תוסבר בסעיף הבא בהקשר לגיבוי והתאוששות.

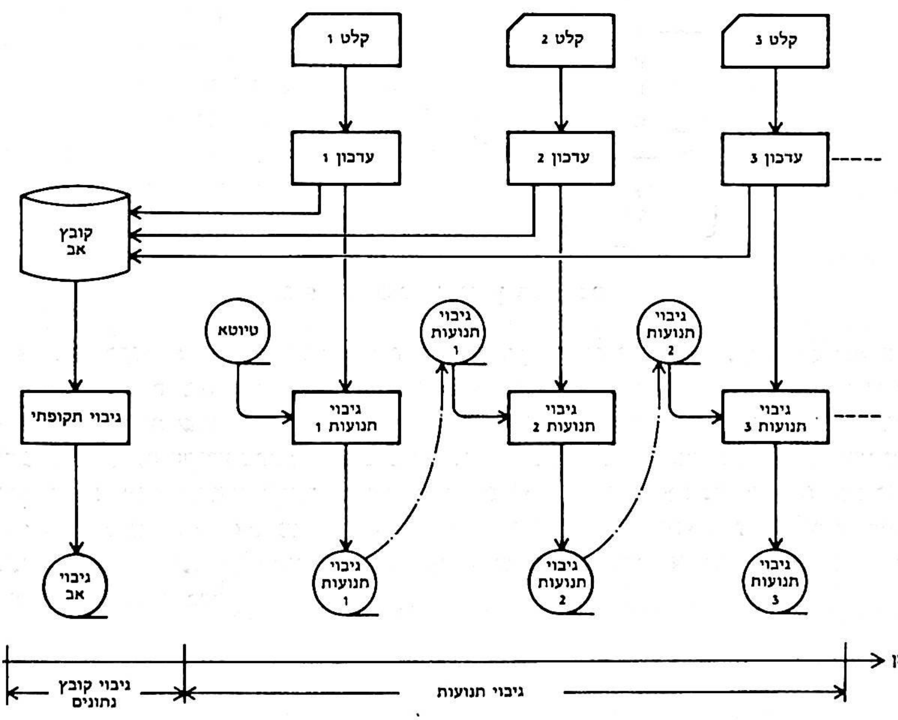
4. גיבוי והתאוששות

בסעיף זה נדון באמצעים השונים המופעלים במערכות DBMS, אשר מיועדים להעלאת זמינות הנתונים. הזמינות מוגדרת כהסתברות שמשתמש ינסה לגשת לבסיס הנתונים והוא יהיה זמין ומוכן לעבודה. שואפים להגיע לזמינות מירבית, כלומר למצב שבו בסיס הנתונים יעמוד לרשות המשתמש בכל עת. את הזמינות רושמים בדרך כלל באחוזים.

תקלות שונות יכולות לקרות: תקלות בתוכניות היישום, הפסקת או נפילת מתח חשמל, תקלות בציוד, טעות הפעלה ועוד. שיתוק בסיס הנתונים עלול לגרום לשיתוק פעילות הארגון כולו. במערכות מידע ממוחשבות לא ניתן להמנע מתקלות שונות שעלולות לפגוע באמינות בסיס הנתונים ועל־כן יש לפעילות הגיבוי וההתאוששות חשיבות עליונה בהבטחת השירות לצרכנים.

4.1 תקלות והתאוששות במערכות שונות

לפני שנמשיך, מן הראוי לסקור את הבעיות במערכות אצווה ובמערכות מקוונות, ולהבין את השוני המהותי שבין שתי מערכות אלו מבחינת גילוי סיבת התקלות והתיקון שלהן.

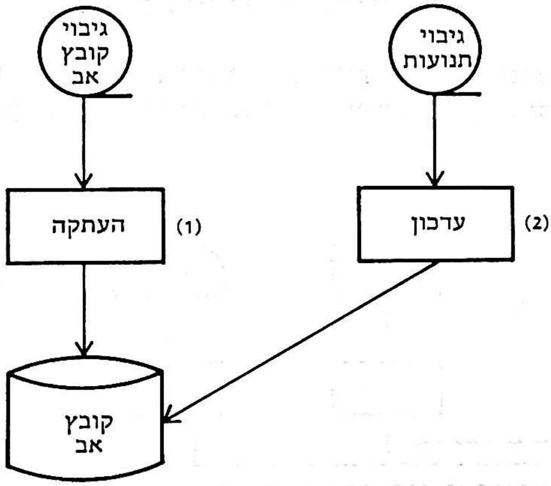


תרשים 4 – 9: גיבוי במערכת אצווה

4.1.1 מערכות אצווה

במערכות הפועלות באצווה (Batch) בעיית הזמינות איננה גדולה מכיון שזמני התגובה שלהן גדולים ואפשר לדחות במידת־מה את מועדי העיבוד עד לתיקון התקלה. השיטה המקובלת להגנה היא **גיבוי** (Backup) **ושחזור** (Restore). על בסיס תקופתי מבצעים **גיבוי של קבצי האב** (Master File Backup) ולאחר כל מעבר עדכון מבצעים **גיבוי של התנועות** (Transaction Backup) שעדכנו את קבצי האב. קובץ הגיבוי הינו העתק מדוייק של קובץ האב נכון לרגע מסויים, או של אוסף התנועות שעדכנו את קובץ האב בכל אחד ממהלכי העדכון. תרשים 4—9 מציג את התהליך.

לאחר תקלה מפעילים את תהליך השחזור שבו מקימים מחדש את קובץ האב מתוך סרט הגיבוי ומעדכנים אותו מחדש עם התנועות הצבורות בקובץ גיבוי התנועות (תרשים 5—9).



תרשים 5 – 9: שחזור קובץ במערכת אצווה

על מנת לקצר את תהליך השחזור ניתן לשמור רק תנועות שעדכנו את הקבצים ולא את התנועות השגויות. בצורה זו אין צורך להפעיל מחדש את תוכניות הקליטה והעדכון הרגילות אלא רק תוכניות שקוראות את התנועות הטובות ומעדכנות את הקובץ בהתאם. לא רצוי להשתמש לעדכון הקובץ בתוכניות שונות. על־כן מעדיפים להשתמש בתוכניות הקלט והעדכון הרגילות גם אם הן עלולות להיות איטיות יותר. אם לא ניתן להריץ את כל התנועות הצבורות בבת־אחת, ניתן לשחזר את כל הריצות הבודדות לפי הסדר שלהן במציאות. תהליך השחזור עשוי להמשך זמן רב, אבל מאחר ובמערכת אצווה זמן התגובה גדול, אלמנט זמן ההתאוששות איננו קריטי בדרך כלל.

4.1.2 מערכות מקוונות

במערכות DBMS המשמשות גם מערכי תקשורת זמן ההתאוששות קריטי מפני שכל היישומים משתמשים בבסיס הנתונים. בזמן ההתאוששות אין אפשרות להפעיל את תוכניות

היישום ולכן יש מחלקות בארגון שאינן מקבלות שירותי מחשב בעת השחזור. גילוי התקלה ותיקונה במערכת DBMS מורכב יותר, הן בגלל ריבוי המשתמשים והן בגלל הקשרים המורכבים בין הנתונים. לדוגמא, גילוי תקלה בסוג רשומה אחד לא ניתן לתיקון רק על-ידי שחזור אותו סוג רשומה מתוך קבצי הגיבוי בגלל הקשרים של סוג הרשומה עם סוגי רשומה אחרים.

למטרות התאוששות לאחר תקלה שומרות מערכות DBMS את כל הנתונים הדרושים כדי להתגבר על התקלה. נתונים אלו משמשים להחזרת בסיס הנתונים לנקודת זמן שבה הוא היה תקין. בזמן תכנון בסיס הנתונים חייב המתכנן להכין את כל תהליכי ההתאוששות הדרושים לכל סוגי התקלות האפשריות. להלן מספר סוגי תקלות אפשריות:

א. תוכנית יישום מסיימת בצורה לא תקינה, אך עד להפסקת עבודתה היא הספיקה לעדכן חלק מבסיס הנתונים. הסיום הלא־תקין שלה גורם שבסיס הנתונים יהיה מעודכן חלקית בלבד ועל כן אינו אמין עתה.
ב. תוכנית יישום מסיימת בצורה תקינה, אבל יש בה שגיאה לוגית וכל העדכונים שביצעה אינם נכונים.
ג. מערכת DBMS מגלה דבר לא תקין ומסיימת את עבודתה, כמו למשל בעת גילוי של מצביע לא תקין או נתוני בקרה לא תקינים.
ד. תקלה בחומרה. הדיסק המכיל את הנתונים אינו ניתן לקריאה או לכתיבה בגלל תקלה פיסית כלשהי (למשל בראש הקריאה/כתיבה או במשטח המגנוט).
ה. נפילה פתאומית של מתח החשמל שעוצרת את כל הפעילות במחשב.
ו. טעות הפעלה אשר גורמת להפסקה בלתי צפוייה של פעילות המחשב או התוכנית.

גילוי סיבת תקלה הנובעת מבעיות תוכנה הוא תהליך מורכב ומסובך יותר מאשר במקרה של תקלת חומרה. גילוי תקלות תוכנה יכול להיות אוטומטי רק כאשר תוכנית היישום מבצעת מספר בדיקות הכרחיות לגילוי מצבים חריגים. לעיתים הזמן שעובר בין אירוע התקלה לבין הגילוי שלה עלול להיות ארוך למדי ועקבות התקלה עלולים להטשטש. מסיבה זו, מערכות DBMS חייבות לשמור נתונים מספיקים כדי לאפשר שחזור גם לאחר תקופת פעילות ארוכה.

4.2 שירותים של מערכות לניהול בסיסי נתונים

רוב מערכות DBMS מציעות כלים לטיפול בבעיות זמינות הנתונים. נסקור את החשובים שבהם.

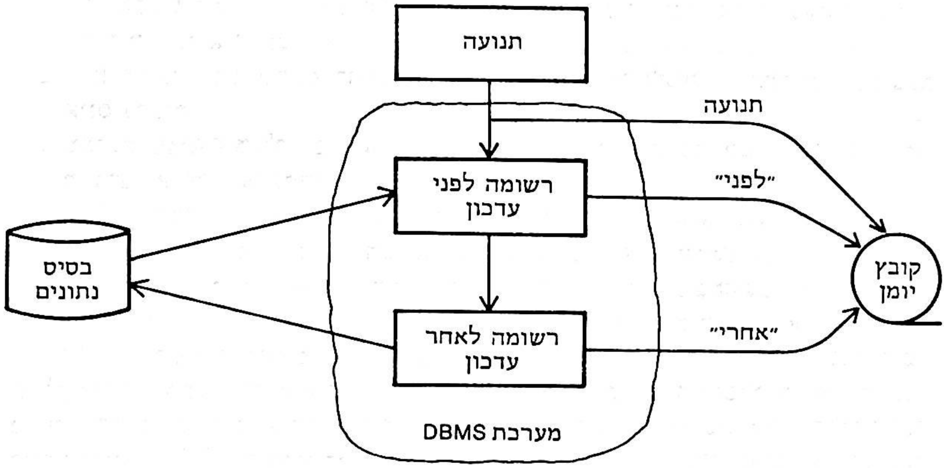
4.2.1 תוכנית שירות לגיבוי (Dump Utility)

תוכניות שירות לגיבוי מאפשרות העתקת בסיס הנתונים לסרט גיבוי. בזמן ביצוע ההעתקה אין אפשרות (לרוב) לתת את השירות לתוכניות היישום ולכן מבצעים אותה בזמן שבסיס הנתונים אינו פעיל, למשל בלילה. העתקת בסיס נתונים נעשית בצורה מיוחדת אשר תאפשר את השחזור לאחר מכן. ההעתקה יכולה להמשך דקות או עשרות דקות כאשר בסיס הנתונים קטן והיא עלולה להמשך שעות אחדות כאשר בסיס הנתונים גדול. משך זמן הגיבוי ישפיע על תכיפותו. משך זמן השחזור שוה ואף ארוך מזמן הגיבוי כי יש לבצע פעולות ניהול שונות במערכת. נראה אם כן ששיטה זו אינה מתאימה להתאוששות מהירה בבסיסי נתונים גדולים.

4.2.2 יומן שינויים (Logging Facility)

המפתח לתהליך ההתאוששות הינו קובץ המכיל את כל השינויים שבוצעו בבסיס הנתונים. לשמירת תמונת מצב הסטורית של בסיס הנתונים יש לשמור את כל התנועות אשר גרמו לשינוי כלשהו. שינויים אלה נשמרים בדרך כלל בסרט מגנטי, בעיקר מסיבות כלכליות ונוחות שמירת הסרטים. מטרת **קובץ היומן** (Log File) היא לאפשר את שחזור בסיס הנתונים תוך שימוש בגיבוי קודם שלו.

קבצי היומן יכולים להכיל את **התנועה עצמה** (Transaction Image), את **מצב הרשומה בבסיס הנתונים לפני העדכון** (Before Image) ואת **מצב הרשומה לאחר העדכון** (After Image). רוב מערכות ניהול בסיסי נתונים מאפשרות למנהל בסיס הנתונים להגדיר את סוגי הרשומות שיש לשמור בקובץ היומן.



תרשים 6 – 9: יצירת יומן שינויים

לפני הרישום של רשומות התנועה מוסיפה מערכת DBMS נתונים נוספים כגון מקור התנועה, שעת העדכון, תאריך העדכון ועוד. בכל פעם שמתבצע עדכון של בסיס הנתונים אובדים חלק מהנתונים שהיו לפני העדכון. כדי לאפשר את החזרת בסיס הנתונים למצבו לפני העדכון יש לשמור בקובץ היומן את מצב הרשומה "לפני". רשומות אלו יכולות לשמש **לביטול השפעת העדכון** (Backout). עם גילוי תקלה ניתן להעתיק מתוך קובץ היומן רק את רשומות "לפני" ולכתוב אותן חזרה לבסיס הנתונים ובצורה זו להחזיר אותו למצב שלפני העדכון. רישום מצב "אחרי" מאפשר שחזור של קובץ לאחר גיבוי או שחזור ללא מעבר דרך תוכניות העדכון הרגילות. הוא גם מאפשר כתיבה של הרשומה בקובץ אם חלה תקלה לפני שהיא נכתבה.

תהליך רישום יומן השינויים ועדכון בסיס הנתונים וכן גם תהליך ההתאוששות צריכים לענות על שלשה מצבים אלו:

— התוכנית הספיקה לקרוא את הרשומות המבוקשות מתוך בסיס הנתונים וקרתה תקלה לפני שבוצעו הרישום ביומן והעדכון (כתיבה) בבסיס הנתונים,
— התקלה קרתה לאחר שבוצע הרישום אך לפני שבוצע העדכון,
— התקלה קרתה לאחר שבוצע הרישום והעדכון.

שימוש ברישום "לפני" ורישום "אחרי" בכל מקרה יגרום לכתיבת כמויות גדולות של נתונים

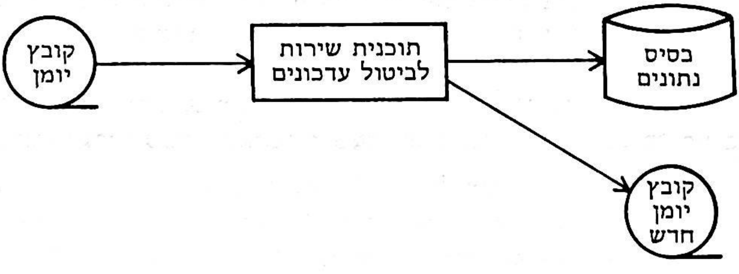
בקובץ היומן כאשר מספר העדכונים רב. בנוסף לכך יש לשקול את ההשפעה על זמן הביצוע של תוכנית היישום כתוצאה ממספר פעולות הכתיבה בדיסק.

4.2.3 נקודת מבדק (Checkpoint Facility)

במערכות המבצעות עיבודים מורכבים וארוכים אין זה רצוי לחזור על כל התהליך החישובי מחדש לאחר שמְתגלית תקלה. כדי לפתור בעיה זו ניתן לכלול נקודות מבדק בתוכנית היישום כדי לצמצם את הזמן הדרוש לביטול העדכונים שבוצעו בבסיס הנתונים לאחר גילוי התקלה. ניתן להוסיף בתוכנית היישום פקודות מיוחדות הגורמות לכתיבת **רשומת מבדק** (Checkpoint Record) בקובץ היומן. רשומה זו מייצגת את מצב המערכת הכולל זכרון, אוגרים, מאגרי תנועות, מצב הקבצים (מאגרים ומספר גוש לרשומה) ומפתח זיהוי המסופק על ידי התוכנית. לאחר גילוי התקלה ניתן לאתר בקובץ היומן את רשומת המבדק האחרונה. תוכנית היישום תקרא את הרשומה הזו או שתוכנית שירות תעשה זאת עבורה. תוכנית היישום תוכל לשחזר על פי רשומת המבדק את המצב שהיה ולהמשיך את עבודתה בבסיס הנתונים מנקודת השחזור.

4.2.4 ביטול עדכונים (Backout Utility)

תוכנית שירות זו מעתיקה מתוך קובץ היומן את רשומות "לפני" ובכך היא מבטלת את השפעת העדכונים של תוכנית היישום. תוכנית השירות קוראת את קובץ היומן בכיוון הפוך, מהסוף להתחלה ויוצרת קובץ יומן חדש אשר אינו מכיל את הרשומות שהשפעתן בבסיס הנתונים בוטלה. קובץ זה ישמש לשחזור המצב הרצוי.



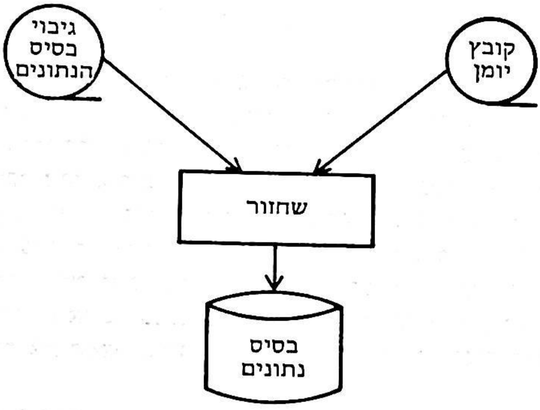
תרשים 7 – 9: ביטול עדכונים

קיימות מערכות שבהן ניתן להפעיל את תוכנית השירות הזו באופן אוטומטי (DTB — Dynamic Transaction Backout) כמו למשל בעת סיום לא תקין של תוכנית יישום. כדי לייעל את תהליך הביטול האוטומטי ניתן לשמור חלק מהשינויים האחרונים בזכרון בנוסף לכתיבתם בקובץ היומן. בעת תקלה יכולה תוכנית ביטול העדכונים לפעול מיד.

4.2.5 שחזור (Recovery Utility)

לצורך שחזור משתמשים בקבצי הגיבוי של בסיס הנתונים ובקבצי היומן של השינויים

שנעשו בבסיס הנתונים. מתוך קבצי הגיבוי מקבלים את מצב בסיס הנתונים ברגע נתון ואליו מעתיקים את כל רשומות "אחרי" מתוך קובץ היומן.



תרשים 8 – 9: שחזור בסיס נתונים

4.3 סיכום

הדיון הארוך בנושא גיבוי, התאוששות ושחזור מפחיד מעט. האמנם חשוף המשתמש לתקלות כה רבות ועליו להשקיע בתכנות, בזמן מחשב ובאחסנה אמצעים כה רבים? התשובה היא שחובה לעשות זאת במערכות מקוונות כדי שמשתמש בודד או תקלה כלשהי לא יגרמו לשיתוק מערך פעילות של משתמשים רבים.

מערכי הרישום של הגיבוי ויומן השינויים מאפשרים התאוששות מהירה, מבחינת מצב בסיס הנתונים ובתוכניות השונות הפועלות במערכת והמשרתות צרכנים ומשתמשים שונים. כל משתמש במסוף שהיה קשור למערכת בעת התקלה יקבל הודעה על התנועה האחרונה שנקלטה בה והנחיות להמשך עבודתו.

האחריות לקיים את הפעילות הרצופה ולזכות באמון המשתמשים מחייבת אחזקת "פוליסת ביטוח". זוהי מערכת גיבוי והתאוששות אשר נלווית למערכת ניהול בסיסי נתונים ומהווה חלק ממנה.

5. בטיחות נתונים

ריכוז כל הנתונים של הארגון במאגר אחד והשימוש הנרחב במערכות תקשורת על ידי משתמשים שאינם בשליטה ישירה של יחידת עיבוד הנתונים עוררו מספר בעיות הקשורות לבטיחות הנתונים (Data Security). בסעיף זה נציג מספר שיטות לאבטחת נתונים אשר מופעלות בעיקר במערכות DBMS. נגדיר תחילה את המושג אבטחת נתונים.

אבטחת הנתונים דנה בנושאים הקשורים במניעת אפשרות הגישה לבסיס הנתונים על־ידי גורמים שאינם מורשים. הגורמים יכולים להיות אנשים בארגון ומחוץ לו ותוכניות יישום הפועלות במערכת. תוכנית הסורקת את קובץ כוח־אדם להפקת דוח סטטיסטי על התפלגות העובדים לפי גיל לא צריכה "לראות" את הנתוני השכר או נתונים רפואיים של אותם העובדים; עובד במחלקת מחסנים אשר יש לו גישה חופשית לכל נתוני המלאי אינו רשאי "לראות" נתונים של מדיניות מחירים והנחות של פריטים במלאי או שאינו רשאי "לראות" נתונים של מחלקות אחרות.

את אבטחת הנתונים אפשר להשיג בשלוש דרכים שונות, בנפרד או בכמה מהן יחד:

— **הרשאת גישה.** מי רשאי לגשת לנתונים (Accessor).
— **סיווג הגישה.** מהן הפעולות המותרות למשתמש (Access Type).
— **נעילת נתונים.** מהם הנתונים שיש למנוע ממשתמש (Data Lock).

5.1 הרשאת גישה

הרשאת גישה לבסיס הנתונים ניתנת למשתמשים ולתוכניות יישום. **שיטת ההרשאה** (User Authorization) מבוססת על מרכיבים שונים אשר תכליתם לאפשר גישה למשתמשים המורשים לכך ולמנוע את הגישה ממשתמשים שאינם מורשים.

ברוב המערכות מבוססת הרשאת הגישה למשתמשים על סיסמא הניתנת לכל אחד מהם. המשתמש המפעיל מסוף חייב להזין את הסיסמא בכל פעם שהוא מתחיל לעבוד במסוף. הסיסמא נבדקת על ידי מערכת ההפעלה, על ידי תוכנת מערכת התקשורת או על ידי המערכת לניהול בסיסי נתונים, בהתאם לעניין. קיימות שיטות שונות לניהול סיסמאות. אפשר לקבוע סיסמאות קבועות או משתנות, סיסמאות לכל יחיד או סיסמא לקבוצת אנשים אשר יכולים לפעול בבסיס הנתונים בעת ובעונה אחת. ככל שרמת ההגנה שרוצים לספק גבוהה יותר, עולה גם עלות השיטה. להלן מספר דוגמאות לשיטות הרשאה למשתמשים.

— **הצמדת משתמש למסוף.** המערכת תאפשר רק למשתמשים מורשים להשתמש במסוף מסויים. לדוגמא עובד במחלקת הנהלת חשבונות לא יוכל להשתמש במסוף אשר שייך למחלקת מחסנים.

— **הגבלת הקבצים במסוף.** ניתן לקבוע שהגישה לקבצים מסויימים או לסוגי רשומה מסויימים מותרת רק ממסופים מורשים. לדוגמא הגישה לקובץ תכנון הייצור אפשרית רק מהמסופים השייכים למחלקה זו.

— **הגבלת התוכניות למשתמש.** לכל משתמש יש רשימה של כל התוכניות שמותר לו להפעיל או להשתמש. הוא לא יוכל להפעיל תוכניות אחרות.

— **הגבלה על בסיס של זמן.** ניתן להגביל את הגישה לקבצים, תוכניות או מסופים לשעות מסויימות. אפשרות מקובלת — למנוע גישה לאחר שעות העבודה הרגילות.

הרשאת הגישה לתוכניות יישום במערכת אצווה, או במערכת מקוונת יכולה להעשות באותה

צורה, אך בדרך כלל משתמשים במקרה זה בשיטות אחרות. ההגבלה של הגישה לנתונים נפתרת כאן על ידי שימוש בתת־הסכימה. תת־הסכימה מיועדת בעיקר להקל על עבודת התוכניתן ועל שליטתו בנתונים אך היא יכולה לשמש גם כאמצעי נוח לאבטחת הנתונים. תוכנית היישום רשאית לגשת רק לנתונים המוגדרים בתת־הסכימה שלה. יש לזכור שבמערכות DBMS תת־הסכימות אינן מוגדרות על־ידי התוכניתן אלא על־ידי מנהל בסיס הנתונים.

סיווג הגישה

סיווג הגישה הינו הרשאה לביצוע פעולות עיבוד מסויימות בבסיס הנתונים.

— **קריאה** (GET): קריאת נתונים לתוכנית היישום.
— **ביטול** (DELETE): ביטול נתונים בבסיס הנתונים.
— **כתיבה** (INSERT): הוספת רשומות חדשות לבסיס הנתונים.
— **שינוי** (MODIFY): ביצוע שינוי כלשהו ברשומה קיימת. לעיתים יש צורך להגביל את השינוי לרמת שדה בתוך הרשומה.

ניתן להרשות לכל משתמש צורת גישה אחת או יותר, כלומר קריאה בלבד, קריאה ושינוי ועוד. בדרך כלל תוגבל הרשאת הביטול עד כמה שאפשר.

נעילת נתונים

— **שדה:** זוהי היחידה הבסיסית ביותר לנעילה מכיון שהיא מכילה עובדה שיתכן שיש למנוע מהמשתמש מלראות. אפשרות אחת היא לא להגדיר את השדה בתת־סכימה. במקרים מסויימים הבעייה מורכבת יותר כאשר האיסור לראות את הנתונים תלוי בתוכן השדה. למשל אם שדה השכר מכיל ערך מסויים ומעלה — הגישה אינה מותרת לכל עובד ביחידת שכר. במקרה זה יש להפעיל שגרה כלשהי שתבצע את הבדיקה.

— **רשומה:** אפשר למנוע גישה של משתמש לרשומה. אם תוכנית אינה מורשית לראות רשומה היא לא תופיע בתת־סכימה שלה.

— **קשר בין רשומות:** במערכת DBMS מאחסנים בנוסף לנתונים גם את הקשרים ביניהם. לעיתים יש למנוע את הגישה לרשומות הקשורות לרשומה מסויימת. לדוגמא לא ניתן לגשת לרשומות הסטוריה רפואית של עובד בכלל, או כאשר הן מכילות נתונים על מחלות מסויימות בלבד.

— **תת־סכימה:** הגישה לתת־סכימה לא תורשה לתוכניות שתת הסכימה לא הוגדרה והורשתה עבורן.

— **סכימה:** הגישה לסכימה לא תותר לתוכניות מסויימות. הגישה לסכימה יכולה להיות מוגבלת לסוג הגישה. תוכניות שונות המייצגות משתמשים שונים יכולות לקרוא את הסכימה אך רק מנהל בסיס הנתונים מורשה לשנות אותה.

נציג לדוגמא סכימה אשר כוללת מספר מרכיבי אבטחת נתונים. הסכימה מבוססת על המלצות CODASYL וכוללת מנגנון של **מנעול** (Lock) **ומפתח** (Key). את המנעול מגדירים בסכימה או בתת־סכימה על־ידי קביעת ערך כלשהו. תת־סכימה או תוכנית שרוצה להשתמש בנתונים צריכה לספק את המפתח למנעול כלומר לדעת את "הערך הסודי". ניתן להגדיר מנעול שונה עבור כל אחד מסוגי הגישה ובכל אחת מהרמות: שדה, רשומה, קשר, תת־סכימה וסכימה.

נעילה של הסכימה

```
SCHEMA NAME IS HOSPITAL.
  PRIVACY LOCK FOR COPY IS ABCD.
  RECORD NAME IS PHYSICIAN.
```

נעילה של הרשומה לפי סוג הפעולה

```
   PRIVACY LOCK FOR DELETE IS AABB,
   PRIVACY LOCR FOR MODIFY IS CCDD,
   PRIVACY LOCK FOR FIND IS EEKK,

   LOCATION MODE IS CALC USING PHYSICIAN-ID.

     01  PHYSICIAN-ID PICTURE IS 9(5).
     01  NAME PICTURE IS X(30).
     01  SPECIALIZATION PICTURE IS X(40).
```

נעילה לשינוי שדה "ההתמחות". השגרה תבצע בדיקות מסויימות

```
     PRIVACY LOCK FOR MODIFY IS DB-PROC-109.
```

בדוגמא זו מופיעות מספר נעילות ברמות השדה, הרשומה והסכימה. רק מי שיודע את **כל** הסיסמאות המתאימות יוכל לגשת לנתונים ולבצע את הפעולה. לדוגמא, תת־סכימה המשתמשת בסכימה תצטרך להגדיר את המפתח בצורה הבאה:

מפתח של הסכימה

```
SUBSCHEMA NAME IS PATIENT-REPORT,
  FOR HOSPITAL SCHEMA
  PRIVACY KEY IS ABCD.
```

6. סיכום

השימוש במערכות DBMS הביא להתפתחות רבה בנושא המערכות המקוונות. התפתחות זו הביאה לשינוי שיטות העבודה בארגון והפכה אותו לתלוי במידה רבה בתפקוד של מערכת המידע הממוחשבת. תלות זו, בנוסף להשקעות הרבות של הארגון בהקמה ובתפעול מערכת DBMS, הביאה לדרישות תפעוליות מיוחדות.

הארגון אשר מפעיל מערכת ניהול בסיסי נתונים מצפה ליתרונות שיושגו ורוצה להקטין ככל האפשר את הסיכון הכרוך בכך. מסיבה זו הושקעו אמצעים רבים בכדי לפתח כלים שונים המאפשרים תפעול מערכות בסיסי נתונים ברמת סיכון נמוכה יחסית. בפרק זה סקרנו את האמצעים הנפוצים והעיקריים המכוונים להשגת מטרה זו.

פרק י

מערכות DBMS —
מגמות וכיוונים לעתיד

1. מבוא

2. מכונת בסיס הנתונים

3. מערכות בסיסי נתונים מבוזרים

4. סיכום

1. מבוא

קצב השימוש ביישומי מחשב הולך וגדל בשנים האחרונות. בין הנושאים שבהם קצב הגידול גבוה במיוחד, נמצא את אחסון ואחזור הנתונים במערכות אצווה ובמערכות מקוונות. כיום, טכנולוגיית מערכות ניהול בסיסי נתונים הינה טכנולוגיה בוגרת עם מספר אלפי מערכות DBMS עובדות לשביעות רצון המשתמשים בהן. בקרב אנשי המקצוע רווחת הדעה שמספר מערכות אלו ילך ויגדל בשנות ה־80. בלי ספק נהיה עדים בתקופה הקרובה למספר שיפורים משמעותיים בתחומים הבאים:

— **רמת ביצועים** (Performance): תכנון רוב מערכות DBMS הפועלות היום מבוסס על טכנולוגיית המחשבים של סוף שנות ה־60 ושנות ה־70. קיימת ציפייה רבה לפריצת דרך בנושא הביצועים של מערכות DBMS ושימוש בטכנולוגיות חדישות, כגון מחשבים מיוחדים לטיפול בבסיסי נתונים, ביזור בסיס הנתונים בין מספר מחשבים שונים וניצול ארכיטקטורה של חומרה של היע״מ ומערכות האחסון.

— **שימושיות** (Usability): רוב מערכות DBMS הקיימות היום מאפשרות **גישה לרשומה אחת** (One-Record-at-a-Time). שיטה זו מחייבת כתיבת תוכניות מחשב מורכבות יחסית לביצוע כל הגישות לבסיס הנתונים בהתאם לסדר הדרוש לעיבוד הנתונים. מערכות המשתמשות במודל הטבלאי הן **יוצאות דופן** בכך שהן מאפשרות גישה אל ״קבוצה״. ההפרדה בין המערכות לפי מודל הנתונים שבו הן משתמשות מקשה על המשתמש כי למטרות מסויימות מתאים מודל הנתונים שנבחר, אך הוא אינו מתאים למטרה אחרת ולכן גורם לסיבוך מיותר בפתרון הבעייה. עדיין לא קיימת טכניקה אחידה ומושלמת לתכנון מבנה הנתונים עבור כל היישומים הצפויים להשתמש במבנה הנתונים. הציפיה היא שבעתיד מערכות DBMS תספקנה מבני נתונים גמישים יותר (כדוגמת מערכות מודל הנתונים הטבלאי) וטכניקות טובות יותר לתכנון מבנה הנתונים.

— **אמינות נתונים** (Data Integrity): ככל שמספר המשתמשים במערכת DBMS הולך וגדל מחמירה בעיית אמינות הנתונים המאוחסנים בבסיס הנתונים. מערכות DBMS חייבות לספק כלים מגוונים להגנה על הנתונים מפני גישות לא מורשות בשוגג או במזיד וכלים להתאוששות מתקלות חומרה או תוכנה. הטכנולוגיות החדישות של בסיסי נתונים מבוזרים גורמות רמת מורכבות גבוהה יותר שבה צריך לטפל. הציפיה היא שמערכות DBMS החדישות תספקה כלים משוכללים לטיפול בבעיות שהוזכרו.

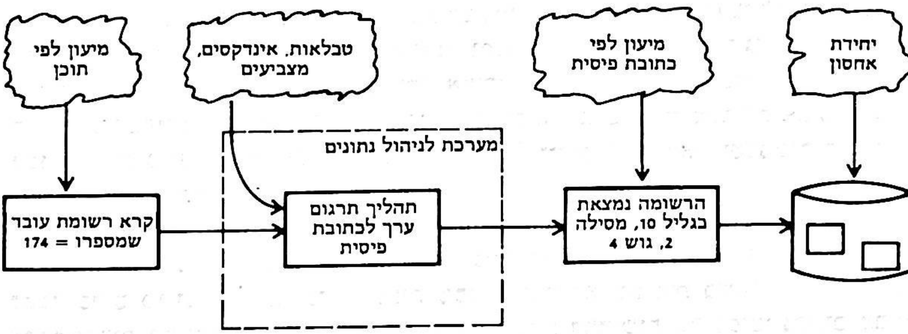
בפרק זה נסקור בקצרה חלק מהטכנולוגיות והטכניקות החדישות אשר צופים את יישומן במערכות ניהול בסיסי נתונים בשנים הקרובות. מן הראוי להזכיר את הדיון במערכות המבוססות על המודל הטבלאי אשר יצאו מכלל ניסויים ובדיקות ועתה משמשות כמערכות תפעוליות וכמערכות שירות למשתמשים בארגונים שונים ולמטרות שונות.

הערה: המונח ״מכונת בסיסי נתונים״ אשר נשתמש בו בהמשך אינו מייצג מכונה מיוחדת שנבנתה למטרה זו בלבד. בפרק זה השתמשנו במונח זה במובנו הרחב והצבענו על כיווני הפיתוח של מערכות חומרה רגילות לעיבוד נתונים כדי לספק שירותים טובים יותר למשתמשים במערכות בסיסי נתונים.

2. מכונת בסיס נתונים

טכנולוגיית המחשבים כפי שהיא מוכרת היום מותאמת היטב לטיפול בבעיות של עיבוד נתונים, ומותאמת פחות לטיפול בבעיות של ניהול נתונים. המחשבים נבנו בתחילה לטפל במהירות וביעילות רבה בחישובים מורכבים ועיבודים מדעיים שונים ולא בבעיות של אחסון ואחזור נתונים. אי התאמה זו נובעת מהעובדה שאחסון ואחזור נתונים מחייבים **מיעון** (Addressing) **לפי תוכן** בעוד שהמחשבים תוכננו למיעון **לפי כתובת פיסית**. לדוגמא, כדי לאחזר רשומת עובד לפי תוכן המפתח שהוא מספר תעודת הזהות למשל יש להציג את הכתובת הפיסית שבה הרשומה מאוחסנת.

המערכות לניהול נתונים משתמשות בטכניקות שונות לביצוע תרגום של התוכן לכתובת, כגון מצביעים, אינדקסים או טבלאות מיוחדות. שיטה זו מוסיפה אלמנט מסויים של חוסר יעילות בניצול המחשב ושטח האחסון. לעיתים טבלאות התרגום השונות דורשות מקום אחסנה רב. גם זמן החישוב הנדרש לתרגום ולחישוב הכתובת הפיסית עשוי להיות משמעותי במקרים מסויימים (מחשב קטן או משתמשים רבים מאד).



תרשים 1 – 10: תרגום תוכן שדה לכתובת פיסית

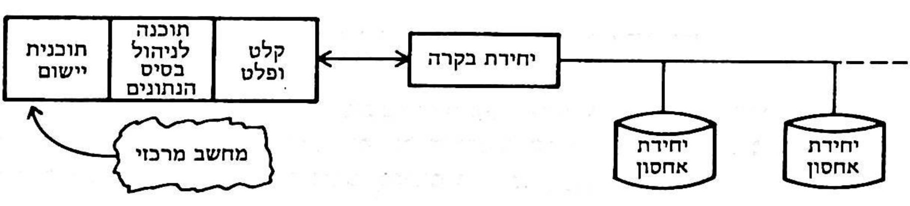
אחת ההתפתחויות המעניינות בשטח המיחשוב היתה הופעת מחשבים מיוחדים המתמחים בנושאים יחודיים. ההתפתחות הגדולה שחלה בשטח תקשורת הנתונים והפעלת המערכות המקוונות היתה לאחר הופעת **יחידות בקרה ממוחשבות לתקשורת ומחשבי תקשורת** (Front-End Communication Processors). הם הורידו את העומס מהמחשב המרכזי וביצעו מטלות מיוחדות הקשורות לניהול התקשורת בין המסופים לבין המחשב המרכזי. יעילות **מחשב יעודי** (Dedicated Computer) נובעת מכך שתכנונו מכוון לתחום יחודי ומוגבל יחסית. מחירו זול ממחיר מחשב המיועד לבצע מגוון של פעולות ולתת שירותים רבים.

חוסר היעילות של המחשבים הקיימים באחסון ואחזור נתונים הנחו את העוסקים בנושא למסקנה שיש לנסות ולפתח מחשבים יעודיים גם לטיפול בבסיסי נתונים. מחשבים אלה יטפלו

בכל ההיבטים הקשורים לניהול בסיס הנתונים ויורידו מעמסה זו מהמחשב המרכזי. מחשבים יעודיים אלה ידועים בשם הכללי **מכונת בסיס נתונים** (DBM — Data Base Machine). מכונת בסיס נתונים הינה שילוב של חומרה ותוכנה המכוונים לעסוק במגוון הנושאים הקשורים לניהול בסיס הנתונים ואין הם עוסקים בפעולות עיבוד נתונים אחרות. מחשבים אלה יעילים במיוחד בפעולות אחסון ואחזור נתונים. פיתוחם עונה על הצורך הגובר של היישומים לאחסן כמויות גדולות של נתונים, לעדכן אותם ולבצע חיפושים מורכבים למטרות אחזור. הטכניקות הקיימות למיעון אינן מתאימות עדיין עבור בסיסי נתונים גדולים (מעל 10^{10} בתים) ולכן השימוש במכונות אלו מוגבל.

בשנים האחרונות חלו התפתחויות בחומרת מערכות המחשבים (יע"מ ודיסקים) אשר דחו את ההכרח לעבור ליחידות עיבוד יעודיות לניהול בסיסי הנתונים, מאידך, מכונות בסיסי הנתונים לא עברו את שלבי הניסויים וההפעלה בנפחי נתונים מוגבלים.

אחד המושגים החשובים ביותר בהבנת טכנולוגיית מכונות בסיסי נתונים הוא **הזכרון האסוציאטיבי**. לאחר הבהרת מושג זה נציג בקצרה חלק מהארכיטקטורות הקיימות ליישום מכונת בסיס נתונים.



תרשים 2 – 10: השיטה המקובלת לניהול בסיסי נתונים

2.1 זכרון אסוציאטיבי

אחד העקרונות החשובים במכונת בסיס נתונים היא האפשרות לגישה לנתונים על פי התוכן ולא על פי הכתובת הפיסית. יחידות אשר מבצעות גישה לנתונים לפי תוכן נקראות **זכרון אסוציאטיבי** (Associative Storage), **זכרון לפי תוכן** (Content-Addressed Storage) או **זכרון חיפוש** (Search Memory).

השיטה הרגילה לגישה לנתונים מנקודת מבט המשתמש היא באמצעות תוכן שדה כלשהו. בדוגמא שהוצגה בתרשים 1—10 הגישה לרשומת העובד היא באמצעות תוכן שדה "מספר־תעודת־זהות". המחשב אינו יכול לבצע חיפוש לפי תוכן אלא רק לפי כתובת פיסית, ולכן מופעל תהליך תרגום כלשהו, שבסופו מתקבל מצביע שמכיל את הכתובת הפיסית של הרשומה

המבוקשת. תהליך התרגום יכול להתבצע על־ידי **רנדומיזציה** (Hashing) או סריקת טבלאות מיוחדות שמקובל לכנותן **אינדקסים.** לפיכך שיטות אלה מחייבות סריקת אינדקסים, ניהול שטחי גלישה, שרשור רשומות ועוד.

הזכרון האסוציאטיבי, בניגוד לשיטה הרגילה, מאפשר גישה ישירה אל הנתונים על־פי התוכן ללא שימוש בכתובות הפיסיות שלהם. מקובל לחלק את הזכרונות האסוציאטיביים לשתי קבוצות עיקריות: זכרון אסוציאטיבי מקבילי וזכרון אסוציאטיבי סדרתי.

— **זכרון אסוציאטיבי מקבילי** (Parallel Associative Storage) יכול לאתר נתונים על פי תוכן, במחזור זכרון אחד הנמדד בסדר גודל של מספר מיקרושניות. הטכנולוגיה הקיימת בזכרונות אסוציאטיביים מקבילים מגבילה את גודל הזכרון למספר מאות או אלפים של מילים. מגבלה זו מונעת יישום השיטה עבור בסיס נתונים ומצמצמת אותה לאחזקת טבלאות מערכת למשל.

— **זכרון אסוציאטיבי סדרתי** (Sequential Associative Storage) מאתר את הנתונים על ידי סריקה סדרתית של הזכרון וביצוע השוואות. הסריקה מתבצעת במהירות רבה עד למציאת הנתונים הדרושים. בטכנולוגיה הקיימת היום משך החיפוש בזכרון של 200 אלף בתים הוא כ־20 אלפיות שנייה. נפח זה אינו מתאים בדרך כלל לניהול בסיסי נתונים אלא לניהול קבצי מערכת ואינדקסים.

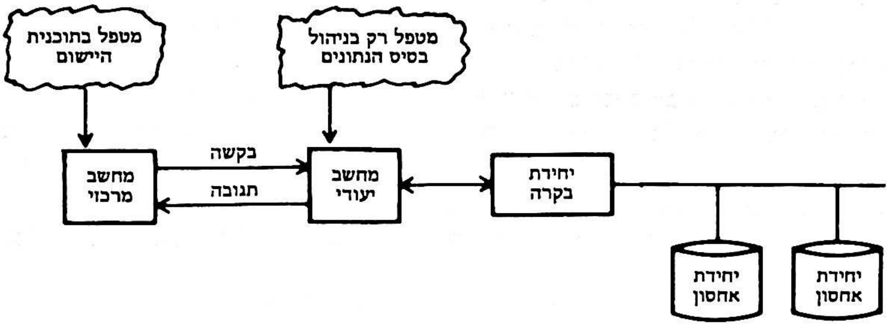
אחד היתרונות הבולטים של הזכרון האסוציאטיבי הסדרתי הוא האפשרות לבצע חיפושים על־פי מפתחות חלקיים, כמו למשל חיפוש רשומת עובד לפי שם משפחתו בלבד ולא לפי שם מלא. בזכרון האסוציאטיבי המקבילי ניתן לבצע חיפושים רק על פי מפתח שלם. יתרון זה נובע מהעובדה שבזכרון האסוציאטיבי הסדרתי מתבצעת סריקה סדרתית על כל הנתונים ולכן ניתן להשוות גם מפתחות חלקיים.

2.2 חלופות לעיצוב מערכות מכונות בסיסי נתונים

קיימות מספר חלופות שונות לעיצוב מערכות DBM ולכל אחת מהן יתרונות וחסרונות. כמה מהחלופות, ובמיוחד אלו המתבססות על היררכיה של זכרונות ויחידות בקרה היקפיות, מופעלות במערכת יבמ לניהול בסיסי נתונים גדולים.

2.2.1 מחשב יעודי התומך במחשב המרכזי (Back-End Processor)

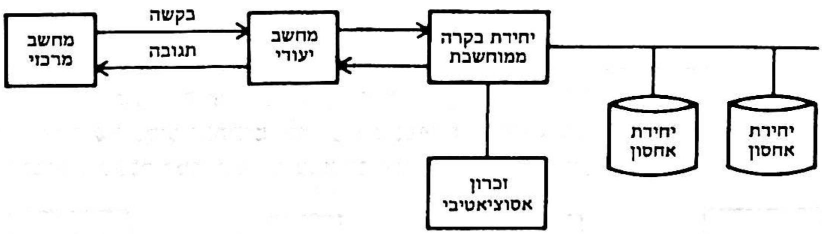
זוהי הארכיטקטורה הישנה ביותר של מערכת DBM אשר יושמה בדרך כלל באמצעות מחשבים קטנים בשיטת Master-Slave. המחשב המרכזי מעביר למכונת בסיס הנתונים את כל הבקשות לגישה לבסיס הנתונים. המחשב היעודי מבצע את כל הפעילויות הדרושות כגון בדיקת הרשאות לגישה לנתונים, ניהול שטחי האחסון של בסיס הנתונים, טיפול בעדכונים בו־זמניים, פעולות קלט ופלט, גישה לנתונים ועריכת הנתונים שנשלפו. בגמר ביצוע הפעולות הדרושות מעביר המחשב היעודי את התשובה אל המחשב המרכזי. התקשורת בין המחשבים מתבצעת באמצעות ערוץ קלט/פלט מהיר אבל ניתן גם לקשור ביניהם באמצעות קוי תקשורת רגילים.



תרשים 3 – 10: מחשב DBM יעודי

2.2.2 יחידת בקרה היקפית ממוחשבת (Intelligent Peripheral Control Unit)

חלק מהפעולות אשר התבצעו במחשב היעודי הועברו אל יחידת הבקרה של יחידת האחסון. יחידת הבקרה מטפלת בפעולות פשוטות החוזרות על עצמן מספר רב של פעמים כמו חיפוש, מיון, התאוששות מתקלות חומרה, מיקום ראשי קריאה/כתיבה וכדומה. בדרך כלל החיפושים מתבצעים בזכרון אסוציאטיבי סדרתי.



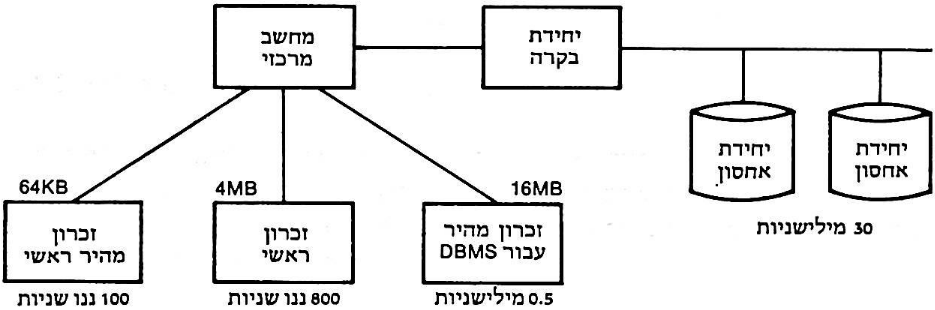
תרשים 4 – 10: יחידת בקרה היקפית ממוחשבת

2.2.3 היררכיה של זכרונות (Storage Hierarchy)

התפיסה של הירככיה של זכרונות קיימת כבר שנים רבות וניתן ליישם אותה גם במערכות

DBM. בשיטה זו בונים מספר זכרונות במקום זכרון מרכזי אחד, כאשר כל אחד מהם הוא בעל מהירות מחזור שונה. בראש ההירקכיה נמצא הזכרון המהיר ביותר, שהוא גם הקטן ביותר, מפני שמחירו הוא הגבוה ביותר. בתחתית ההירקכיה נמצאת יחידת האחסון בגישה ישירה, הדיסק, שנפח האחסון שלו גדול אבל מהירותו איטית באופן יחסי לזכרון.

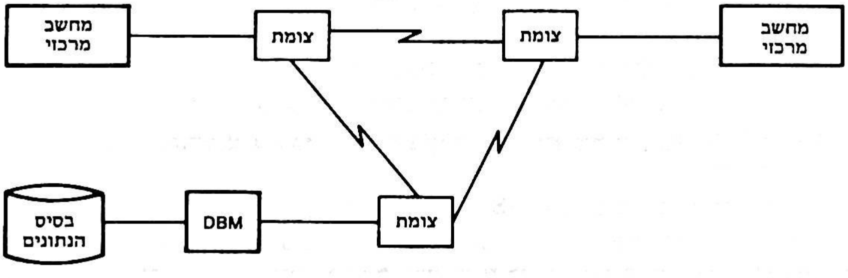
בשיטה זו בונים במחשב היעודי זכרון מהיר יחסית אבל בעל כושר אחסון בינוני. הרעיון של בניית זכרון זה נבע מכך שמחקרים שונים הראו שביישומים רבים רוב הגישות **לבסיס הנתונים מתרכזות באותה סביבה פיסית** (Locality of Data Reference). כלומר יש הסתברות גבוהה לשימוש בנתונים בהם כבר השתמשנו או בנתונים סמוכים להם. נתונים בזכרון המהיר מנוהלים על בסיס של **אחרון בשימוש** (Last Used), כלומר כאשר יש להכניס נתונים חדשים לזכרון המהיר הם נכתבים על הנתונים שלא היו בשימוש זמן רב ביותר.



תרשים 5 – 10: מחשב מרכזי עם הירקכיה של זכרונות

2.2.4 רשת מחשבים (Network)

בשיטה זו המחשב המרכזי קשור למספר צמתים ברשת תקשורת כאשר באחד מהם נמצאת מכונת בסיס הנתונים. היתרון של שיטה זו הוא בכך שמספר מחשבים מרכזיים יכולים לגשת להעתק אחד של בסיס הנתונים ולמנוע את כפילות הנתונים. במערכת כזו קיימות בעיות של ניהול ובקרה מעבר למה שמוכר במחשב אחד שמשרת צרכנים רבים.



תרשים 6 – 10: רשת מחשבים עם מכונת בסיסי נתונים מרכזית

3. מערכות בסיסי נתונים מבוזרים

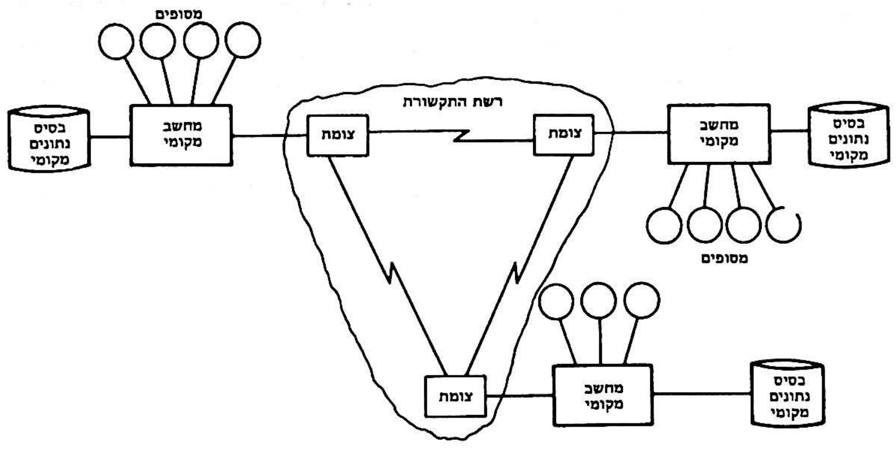
מערכת בסיסי נתונים מבוזרים (DDBS — Distributed Data Base Systems) מאפשרת לאחסן את הנתונים במספר אתרים שונים בתוך רשת מחשבים. למרות האחסון במספר אתרים שונים זהו עדיין בסיס נתונים **אחד** שהגישות אליו מבוקרות ונעשות בצורה אחידה. העניין הרב במערכות בסיסי נתונים מבוזרים נובע מכך שהדבר יכול להוות פתרון טוב לארגונים גדולים המבוזרים גיאוגרפית, אך רוצים לנהל את הנתונים בצורה אחידה בבקרה מרכזית. המורכבות של הבקרה והעלות הגבוהה באמצעים ובקוי תקשורת מהירים הנדרשים למטרה זו מונעים לפי שעה את התפתחות השיטה.

היתרונות של מערכות ניהול בסיסי נתונים מבוזרים הם:

— ניתן לאחסן את הנתונים קרוב למקום בו הם נוצרים או דרושים במידה הרבה ביותר, ובכך להגדיל את מהירות הגישה ולהקטין את עלות תקשורת הנתונים. בשנים האחרונות היתה ירידה יחסית גדולה יותר בעלות העיבוד ואחסון הנתונים לעומת הירידה היחסית במחירי תקשורת הנתונים.

— מערכת בסיסי נתונים מבוזרים הינה אמינה יותר מכיון שבסיס הנתונים אינו נמצא ומנוהל על ידי מחשב אחד אלא על ידי מספר מחשבים. תקלה באחד מהם אינה מפסיקה בהכרח את פעולות שאר המחשבים ותפעול בסיסי הנתונים המקומיים שלהם.

— אפשר לשלוט טוב יותר על נפח בסיס הנתונים ולהתאימו לצרכים המשתנים.



תרשים 7 – 10: בסיס נתונים מבוזר

קיימים נסיונות ראשונים להפעלת מערכות בסיסי נתונים מבוזרים ואין ספק שבעתיד נראה שימוש רב יותר על ידי ארגונים מבוזרים מבחינה גיאוגרפית אשר ינצלו את היתרונות שבשיטה זו. עם זאת, מערכות בסיסי נתונים מבוזרים מציבות מספר בעיות שטרם נפתרו במלואן. נציג אחדות מהן:

— **סינכרוניזציה בעדכונים:** יש צורך בפיתוח מנגנונים יעילים לבקרה על פעולות העדכון של בסיס הנתונים כדי שתישמר אמינותו. בעיה זו מורכבת יותר כאשר נתונים זהים מאוחסנים במספר אתרים שונים, כלומר קיימת כפילות נתונים.

— **ניהול הסכימה:** במערכות בסיסי נתונים קיימת סכימה אשר מתארת את כל הנתונים בבסיס הנתונים ואשר יש לפנות אליה לקבלת נתונים על מנת לבצע את הפעולות הנכונות. במערכת מבוזרת קיימת הבעיה היכן לאחסן את הסכימה וכיצד יש לגשת אליה.

— **שאילתות לנתונים מבוזרים:** הנתונים מאוחסנים במספר אתרים שונים ולכן צריכה השאילתא לעבד לעיתים נתונים מבוזרים. לביצוע העיבוד עליה לבקש לשדר אליה את כל הנתונים הדרושים או לנסות ולעבד אותם בו־זמנית במספר מחשבים שונים, ולבקש שידור של הסיכום או התוצאה.

— **טיפול בתקלות:** כאשר קורית תקלה באחד הצמתים של בסיס הנתונים המבוזר מתעוררות בעיות כיצד להמשיך ולבצע את העדכונים בשאר הצמתים ולשמור על אמינות בסיס הנתונים.

קיימות מספר חלופות ליישום מערכות בסיסי נתונים מבוזרים. אם ניתן לפצל את בסיס הנתונים לחלקים זרים אזי בכל מחשב מקומי יהיה העתק של הנתונים שלו בלבד. גם התנועות וגם הנתונים הם מקומיים בלבד וכל מחשב "מכיר" רק את בסיס הנתונים שלו. אם התנועות במחשב אחד מתייחסות לנתונים המאוחסנים במחשב אחר אזי כל מחשב ברשת צריך "להכיר" את הנתונים שבמחשבים האחרים. קיימות מספר אפשרויות לטיפול במצב זה:

— נאחסן העתקים של כל בסיס הנתונים במחשבים השונים. אם מעדכנים נתון כלשהו במחשב אחד, מחשב זה יוצר תנועות עדכון לכל שאר המחשבים ברשת.

— נאחסן את הנתונים בצומת בה הם דרושים במידה הגבוהה ביותר. תנועות המתייחסות לנתונים שאינם מקומיים יועברו למחשב מרכזי המטפל בכל התנועות האלה או למחשב המאחסן את הנתונים ומעדכן אותם.

4. סיכום

ההתקדמות שחלה בשנים האחרונות בטכנולוגיה של יחידות האחסון עדיין לא סגרה את הפער שבין ׃ ׳כרון המרכזי המהיר לבין יחידות האחסנה המשניות האיטיות יחסית. יש לצפות שמכונות בכ ׳סי הנתונים והשימוש בזכרונות אסוציאטיביים יגדילו במידה רבה את ישימות המערכות ל ׳יהול בסיסי נתונים.

הזכרון האסוציאטיבי קיים כבר משנות ה־60 בהן נבנו מספר מערכות מחקריות. עם השיפורים **בטכנולוגיית המזעור** (LSI) התעורר מחדש המחקר בנושא זה כאשר המטרה היא בניית זכרונות בגודל מספיק ובעלות כזאת שיצדיקו את השימוש גם עבור בסיסי נתונים גדולים מאד. מחקרים שונים מלמדים שטכנולוגיית הדיסקים המגנטיים תשלוט עוד מספר שנים רב תוך שיפורים מתמידים במהירות הגישה, צפיפות האחסון, אמינות ועוד. השיפורים יביאו לעליה מתמדת בנפח האחסון ולירידה בעלות האחסון.

בתחום התוכנה והמישק בין בסיס הנתונים והמשתמש נהיה עדים גם כן למספר שיפורים, כאשר המטרה המרכזית הינה פישוט ואיחוד המודלים של הנתונים הקיימים. המגמה בכיוון זה מזכירה את התהליכים שהיינו עדים להם בשטח שפות התכנות. דוגמא לכך הן העבודות והמחקרים המתבצעים בנושא **ארכיטקטורה אחידה של בסיסי נתונים** (Common Data Base Architecture). ארכיטקטורה זו צריכה להיות מסגרת למערכות השונות הקיימות היום ולאפשר תרגום ממודל נתונים אחד לשני, תרגום של פקודות לטיפול בנתונים ממודל אחד למשנהו וכדומה. נראה שיישום ארכיטקטורה כזאת תאפשר שימוש במודלים הנוחים יותר למשתמש ולכן קלים יותר ליישום ולשימוש. עליה להיות בעלת עוצמה רבה יותר מכיון שהיא משתמשת ביתרונות של כל אחד מהמודלים הקיימים.

מתוך הנחה שכל השיפורים במישור הפיסי של אחסון הנתונים ובמישור הלוגי של השימוש בנתונים אכן יתקיימו, נהיה עדים לשימוש גובר במערכות בסיסי נתונים תוך עליה מתמדת ברמת הביצועים וירידה במחירים.

נסכם את החלק הראשון של הספר בכך שנאמר שהמערכות לניהול בסיסי נתונים עדיין לא מיצו את מלוא הפוטנציאל הגלום בהן, והעתיד צופן בחובו ניצול נוסף של פוטנציאל זה.

חלק שני

מערכות DBMS מסחריות

1. IMS/DL/1
2. IDMS
3. TOTAL
4. ADABAS
5. SQL/DS

בפרקים הבאים תוצגנה מספר מערכות מסחריות אשר נפוצות ומוכרות בארץ. כל אחת מהן מייצגת מדל נתונים שונה או גישה שונה לטיפול באותו מדל נתונים. אין כל כוונה לציין בכך שרק מערכות אלו קיימות בשוק או שהן עדיפות באיזו צורה שהיא על מערכות אחרות שלא הוזכרו.

— המודל ההירככי מיוצג על ידי מערכת IMS של חברת יבמ,
— המודל הרשתי מיוצג על ידי מערכת IDMS של חברת Cullinet ומערכת TOTAL של חברת Cincom,
— המודל הטבלאי מיוצג על ידי המערכות SQL/DS של חברת יבמ,
— מערכת ADABAS של חברת Software AG מייצגת את המערכות המשתמשות בקבצים מהופכים.

המטרה העיקרית בהצגת המערכות הינה **תיאור היישום של מודל הנתונים והשפה שבו הוא משתמש להגדרת הנתונים ולטיפול בהם.** התיאור כולל את המאפיינים העיקריים אשר תומכים בהסבר התיאורטי שניתן בפרקים קודמים, ומאפשר למשתמש להכיר באופן כללי את התכונות של המערכות השונות. לשם החלטה בדבר בחירת מערכת ניהול בסיסי הנתונים לארגון, מעבר לשלב ההחלטה על מודל הנתונים הרצוי, יש לפנות לספרות המקצועית של היצרנים ולקבל את המידע הדרוש במלואו.

יצרני התוכנה מכריזים על עדכונים ועל מהדורות חדשות של מוצרים קיימים. כתוצאה מכך יתכן שיחולו אי התאמות בין תיאורי המערכות בספר זה לבין התוכנה המסופקת על ידי היצרנים, אך בכל מקרה, לא יחול שינוי במודל הנתונים הבסיסי ובשפה הבסיסית של מערכת התוכנה.

פרק יא

IMS —
Information Management System

1. מבוא
2. מושגי יסוד
3. מבנים פיסיים
4. קשרים לוגיים
5. שפה להגדרת נתונים
6. שפה לטיפול בנתונים
7. סיכום

1. מבוא

מערכת IMS/VS הינה מערכת מודל הירכי לניהול בסיסי נתונים אשר פותחה על ידי חברת יבמ והיא בין המערכות הותיקות הקיימות בשוק. חברת יבמ מספקת שתי מערכות ניהול בסיסי נתונים המבוססות על המודל ההירכי: מערכת IMS/VS ומערכת DOS/VS—DL/1. מערכת IMS/VS המיועדת למערכות ההפעלה OS/VS1 ו־MVS נותנת שירותי ניהול בסיס נתונים וניהול רשת תקשורת לשם גישה לבסיס הנתונים באמצעות מסופים, ובכך היא **מערכת בסיס נתונים ותקשורת נתונים** (DB/DC — Data Base/Data Communication). רכיב DB של המערכת נקרא DL/1 והוא אשר מכיל את **מודל הנתונים** ואת **שיטות הגישה אל בסיס הנתונים.** עבור מערכת ההפעלה DOS פותחה מערכת DL/1 עצמאית אשר דומה, אם כי לא זהה, למרכיב DB של מערכת IMS/VS. במערכת ההפעלה DOS הגישה לבסיס הנתונים באמצעות מסופים מבוצעת בעזרת תוכנת התקשורת CICS. בהמשך נתאר את מודל הנתונים DL/1 אשר מיושם במערכת IMS/VS.

במערכת IMS/VS חמישה רכיבים עיקריים:

— **תוכניות היישום** אשר נכתבות על ידי תוכניתני היישום לפתרון בעיות עיבוד הנתונים בארגון. תוכניות אלה משתמשות במישק סטנדרטי לשם תקשורת עם שאר מרכיבי התוכנה. התוכניות משתמשות בפקודת CALL עם אוסף של פרמטרים כדי להפעיל את שאר מרכיבי IMS/VS.

— **בסיס הנתונים** המכיל את הנתונים. המערכת משתמשת במודל נתונים הירכי משוכלל המאפשר לחבר הירכיות שונות ביניהן. הגישה לבסיס הנתונים מתבצעת באמצעות רכיב DL/1 במערכת.

— **מערכת DL/1** הינה אוסף של תוכניות מחשב המתווכות בין תוכניות היישום לבין בסיס הנתונים.

— **טבלאות בקרה** (DL/1 Control Blocks) המשמשות כמישק בין תוכניות היישום לבין מרכיב DL/1 ומתארות את מבנה בסיס הנתונים. הטבלאות נבנות ומתוחזקות על־ידי מנהל בסיס הנתונים (DBA).

— **מערכת התקשורת** הינה אוסף של תוכניות מחשב המאפשרות לתוכנית היישום להתקשר למסופים. מערכת התקשורת גם מנהלת את רשת התקשורת בכך שהיא מפקחת על הרשאת הגישה של המשתמשים השונים, מכילה מערכות שחזור ובקרה ועוד. בהמשך לא נדון ברכיב התקשורת של מערכת IMS/VS.

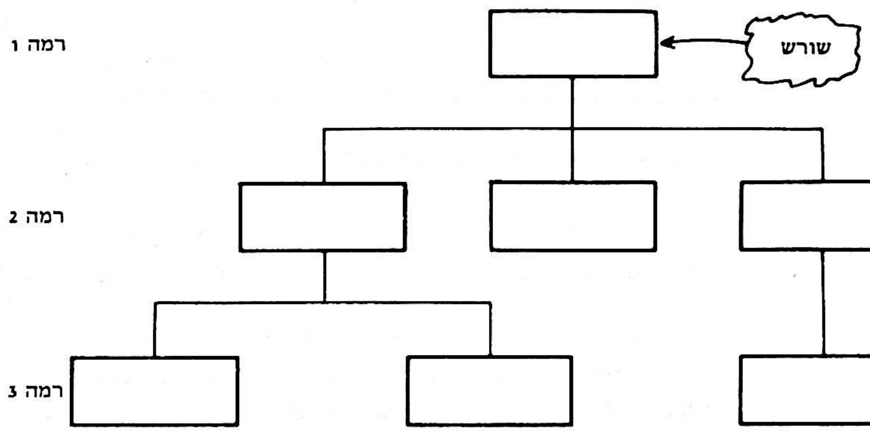
2. מושגי יסוד

במערכת DL/1 נהוגים המושגים הבאים:

— **שדה** (Field): יחידת הנתונים הבסיסית והקטנה ביותר במערכת. לכל שדה יש שם סימבולי אשר מאפשר לתוכניות המערכת גישה אליו.

— **סגמנט** (Segment): הסגמנט הוא אוסף של שדות בעלי שייכות לוגית והוא יכול להיות בעל אורך קבוע או משתנה. קיימת אי תלות בין תוכנית היישום לבין צורת האחסון הפיסי של הסגמנט. אי התלות מושגת על־ידי הגדרת הסגמנטים מחוץ לתוכנית היישום ופנייה אל הסגמנט בצורה סימבולית. ניתן להגדיר עד 255 **סוגי סגמנטים** (Segment types) שונים במסגרת בסיס נתונים אחד. לכל אחד מסוגי הסגמנטים יכול להיות מספר בלתי מוגבל של **מופעים** (Segment Occurrences). ההגבלה היחידה היא שטח האחסנה של בסיס הנתונים.

— **רשומה** (Data Base Record): הרשומה היא אוסף הירכי של סגמנטים אשר יכול להכיל עד 15 רמות הירכיות. החלוקה לסגמנטים ולרמות הירכיות נקבעת על פי דרישות היישומים השונים. הטיפול בניהול המקום הפיסי הדרוש לאחסון המבנה ההירכי הוא בידי DL/1 ו"שקוף" מבחינת המשתמש. במבנה ההירכי הסגמנט הנמצא בראש המבנה יקרא **סגמנט השורש** (Root Segment) והוא מכיל את **המפתח** המזהה את הרשומה כולה.



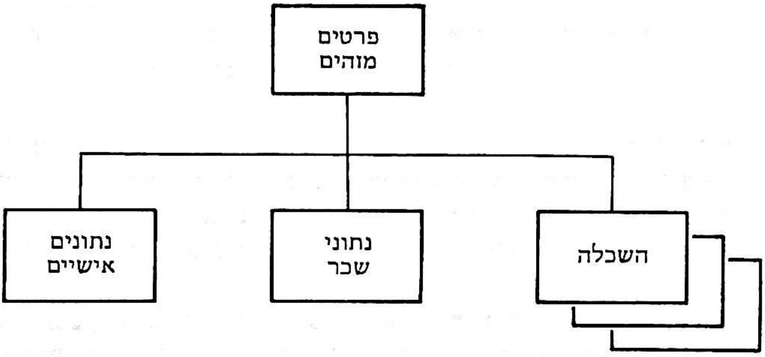
תרשים 1 - 11: מבנה הירכי של רשומה

בקבצים סדרתיים (Sequential Files) מכילה הרשומה את כל הנתונים ברצף, **במבנה רשומה שטוח** (Flat Model) כפי שמוצג בתרשים 2—11.

השכלה	נתוני שכר	נתונים אישיים	פרטים מזהים
1 2 3			

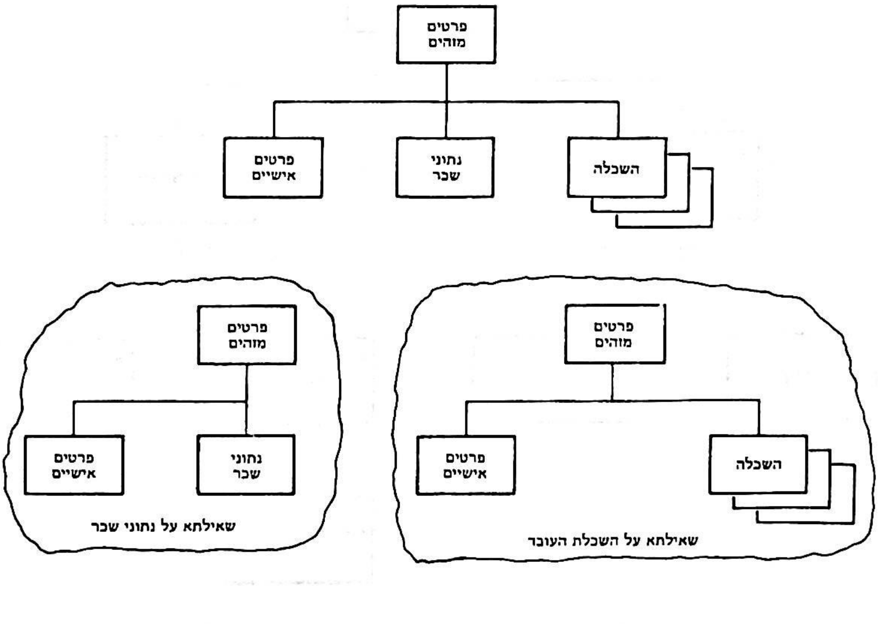
תרשים 2 - 11: מבנה שטוח של רשומת כוח אדם

ב־DL/1 הופך מבנה שטוח זה למבנה הירכי הבנוי מסגמנטים שונים כפי שמוצג בתרשים 3—11.



תרשים 3 - 11: מבנה הירכי של רשומת כוח אדם

חלוקת הרשומה לסגמנטים מאפשרת בין השאר אבטחת הנתונים מפני שימושים בלתי מורשים בכך שהיא מציגה לכל משתמש רק את הנתונים הדרושים לו. המערכת מאפשרת להגדיר לכל משתמש אילו סגמנטים מותר לו לראות או לעדכן. תוכנית היישום יכולה לפנות לסגמנטים אלו בלבד והיא לא תורשה לגשת לסגמנטים אחרים. לדוגמא יש לכתוב שתי שאילתות, אחת להצגת הנתונים האישיים של העובד ונתוני השכר שלו, והשניה — להצגת הנתונים האישיים של העובד ונתוני ההשכלה שלו. לשתי תוכניות היישום עשויה להיות הרשאה שונה לגישה לבסיס הנתונים. מערכת DL/1 מאפשרת להגדיר לכל אחת מהן את המבנה הלוגי שהיא "תראה".



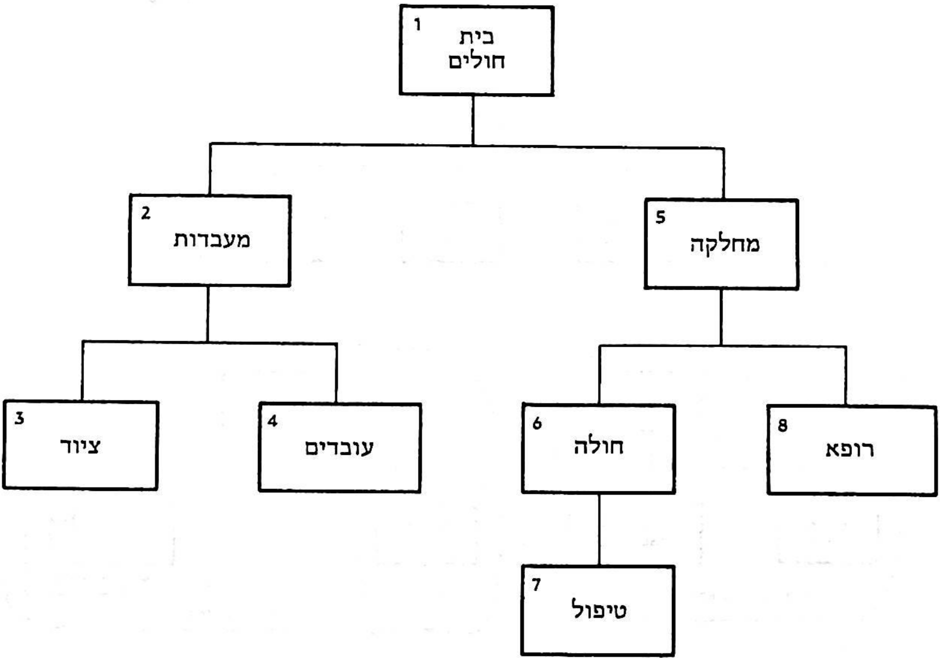
תרשים 4 - 11: שני מבנים לוגיים המבוססים על אותו בסיס נתונים

מספר הסגמנטים ברשומה אחת משתנה בין הרשומות השונות גם כפונקציה של הזמן. ברגע נתון לרשומת עובד אחת יכולים להיות קשורים 4 סגמנטים של נתוני השכלה בעוד שלרשומות עובד אחרת קשורים רק 2 סגמנטים כאלו. במשך הזמן יכולים להתווסף סגמנטים כאלה לכל אחד מהעובדים בהתאם לקורסים או הכשרה שיקבלו.

— **מפתח** (Sequence field): **לכל** רשומה בבסיס נתונים יש מפתח חד־משמעי המזהה אותה. מפתח זה הינו אחד מהשדות **בסגמנט השורש**. גם לסגמנטים התלויים בסגמנט השורש יכולים להיות שדות מפתח אולם הגישה אליהם אפשרית רק לאחר ידיעת המפתח של סגמנט השורש. המפתחות של סגמנטים תלויים אינם חייבים להיות יחודיים.

— **קשרים בין סגמנטים**: מקובל להשתמש בביטוי **הורה** ו־**ילד** או **אב** ו־**בן** לתיאור הקשרים בין הסגמנטים של אותה רשומה. סגמנט שקשורים אליו אחד או יותר סגמנטים ברמה נמוכה יותר יקרא **סגמנט אב** או **הורה** (Parent segment). סגמנט הקשור לסגמנט אחר ברמה גבוהה ממנו יקרא **סגמנט בן** או **ילד** (Child segment). לדוגמא בתרשים 5—11, סגמנט 1 הוא סגמנט האב של סגמנטים 2, 3, 4, 5, 6, 7, 8. סגמנט 2 למשל הוא סגמנט בן של סגמנט 1. מושג האב/בן הינו מושג יחסי מכיון שסגמנט בן יכול להיות בעצמו סגמנט אב. לדוגמא, סגמנט 2 שהינו בן של סגמנט 1 הינו סגמנט אב של סגמנטים 3, 4.

מקובל לכנות את כל המופעים של סגמנטים מאותו סוג הנמצאים תחת אותו סגמנט אב **סגמנטים תאומים** (Twin Segments). לדוגמא, לכל המופעים של סגמנטים בן מסוג 4 המופיעים מתחת למופע מסויים של סגמנט אב מסוג 2 הם סגמנטים תאומים.



תרשים 5 - 11: מבנה היררכי

— **סדר סריקת המבנה ההירככי** (Tree Traversal) על ידי מערכת DL/1 לשם איתור רשומה או סגמנט נעשה בשיטה קבועה. השיטה נקראת **למעלה-למטה, שמאל-ימין** (Top-Down, Left-Right). בתרשים 6—11 נקבעו מספרי הסגמנטים לפי סדר הסריקה של המבנה ההירככי.

11←10←9←8←7←6←5←4←3←2←1

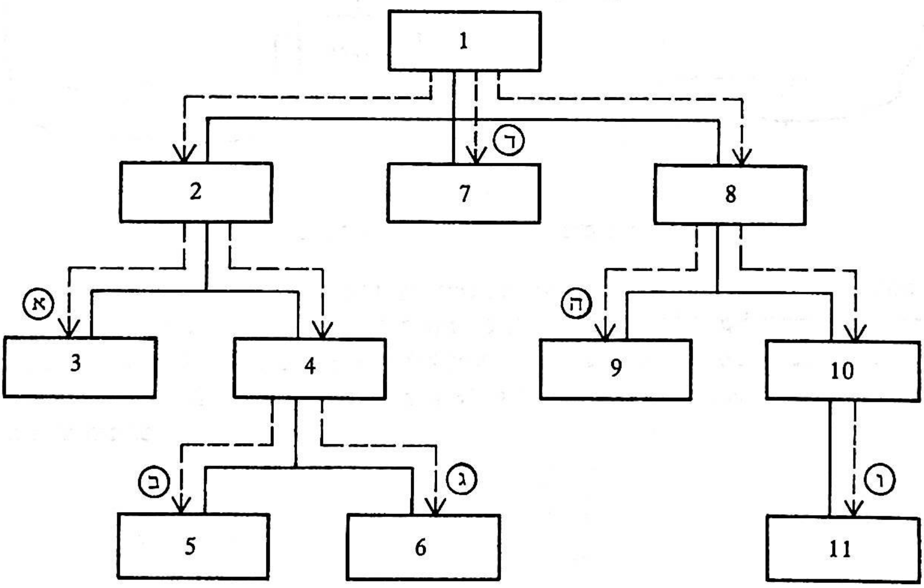
בתרשים 5—11 סומנו גם־כן הסגמנטים לפי סדר הסריקה.

— **מסלול הירככי** (Hierarchical Path): המסלול ההירככי הוא אוסף הסגמנטים שיש לסרוק על מנת להגיע **לסגמנט יעד** כלשהו. מערכת DL/1 מאפשרת גישה לכל אחד מהסגמנטים במבנה הרשומה באמצעות מסלול הירככי אחד. המסלול ההירככי המוביל לסגמנט 6, למשל, בתרשים 6—11 הוא:

6←4←2←1

ברשומה המוצגת בתרשים 6—11 אפשריים המסלולים ההירככיים הבאים:

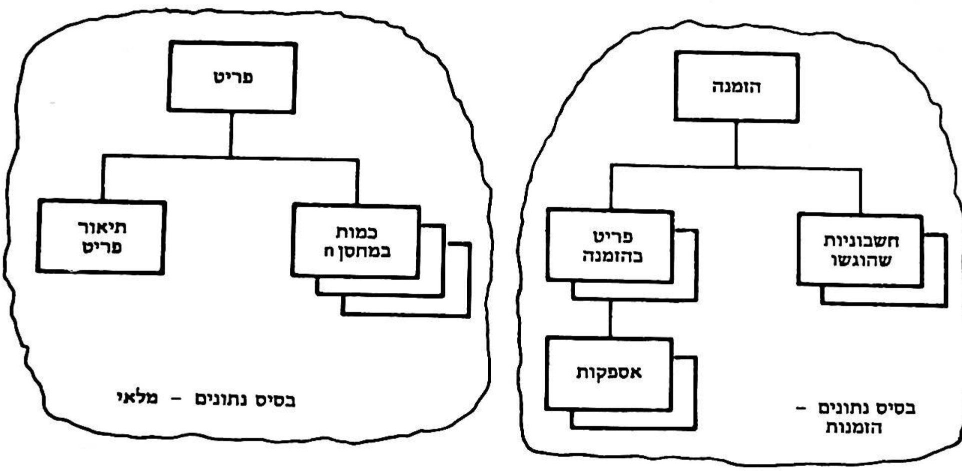
א. 3←2←1
ב. 5←4←2←1
ג. 6←4←2←1
ד. 7←1
ה. 9←8←1
ו. 11←10←8←1



תרשים 6 - 11: מסלולים הירככיים ברשומה

אוסף המפתחות של הסגמנטים המרכיבים מסלול היררכי הוא **מפתח משורשר** (Concatenated Key). כדי להגיע לרשומה 3 למשל, יש להכיר את המפתחות של סגמנט השורש, סגמנט 2 וסגמנט 3.

— **בסיס הנתונים** (Physical Data-Base): בסיס הנתונים מכיל את המופעים של כל הרשומות מאותו סוג, כלומר בעלות אותו מבנה הירכי. המבנה ההירכי של כל מופע הוא שונה בגלל קיום או אי־קיום של סוגי הסגמנטים השונים ברשומה. תרשים 7—11 מציג שני בסיסי נתונים שונים, אחד מכיל את רשומות המלאי והשני את רשומות ההזמנות. בסיס הנתונים "מלאי" למשל, כולל פריטים שונים אשר נמצאים במלאי במחסנים השונים. ברשומות הפריט השונות יהיה מספר שונה של סגמנטים "כמות במחסן" בהתאם למספר המחסנים שבהם מאוחסן הפריט.

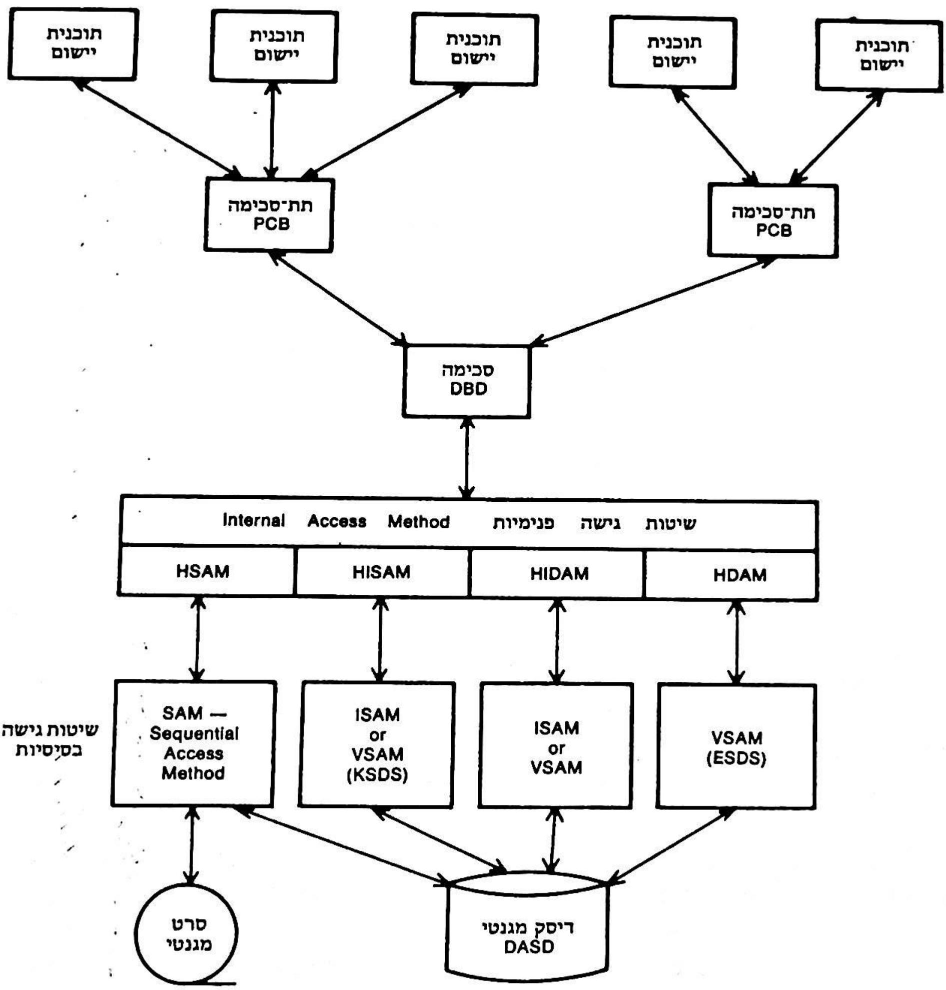


תרשים 7 - 11: שני בסיסי נתונים ב־DL/1

בתוך בסיס הנתונים אין הסגמנטים של הרשומה מאוחסנים באופן רציף, ועל־כן מערכת DL/1 משתמשת במצביעים כדי לקשור את הסגמנטים למבנה הלוגי של הרשומה. קיימת אפשרות להגדיר מספר רב של בסיסי נתונים פיסיים כאשר כל אחד מהם מכיל רשומה אחת או יותר, וכל אחד מהם יכול להשתמש בשיטות שונות לאחסון נתונים. ניתן לקשור בסיסי נתונים פיסיים ביניהם.

3. מבנים פיסיים

מודל DL/1 מאפשר מגוון של שיטות גישה לבסיס הנתונים אשר מגדירות גם את צורת האחסון הפיסית של הנתונים. תוכנית היישום בלתי תלויה בשיטת האחסון מכיון שהיא עובדת תמיד עם מבנה היררכי לוגי. שיטה זו מאפשרת מעבר מצורת אחסון אחת לשניה משיקולי יעילות למשל, מבלי שהדבר ישפיע על תוכנית היישום. בסעיף זה נציג ארבע **שיטות גישה פנימיות של המודל** (Internal Model Access Methods). שיטות אלו מבוססות על שיטות הגישה המקובלות במערכות יבם.

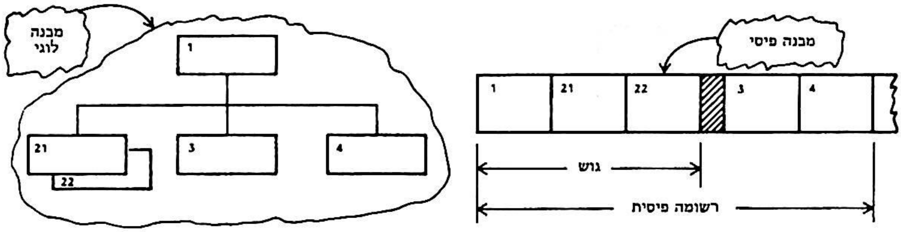


תרשים 8 – 11: שיטות גישה פנימיות ובסיסיות ב־IMS

3.1 HSAM — Hierarchical Sequential Access Method

שיטה זו משתמשת בארגון הסדרתי של הרשומות. הסגמנטים מאוחסנים ברצף בהתאם למבנה ההירכי בגושים בגודל קבוע, ורשומה אחת יכולה לגלוש מגוש אחד לשני. שיטת אחסון זו אפשרית במתקני אחסון לגישה ישירה ובסרטים מגנטיים. השיטה אינה מאפשרת הוספה או ביטול סגמנטים לאחר בניית הרשומה.

קיימת אפשרות להגדיר **מבנה סדרתי פשוט** (SHSAM—Simple HSAM) אשר מכיל סגמנט שורש בלבד, אשר מוכר כקובץ סדרתי רגיל. שיטת הארגון הסדרתית HSAM משמשת בעיקר לשם העתקות גיבוי של בסיס הנתונים.



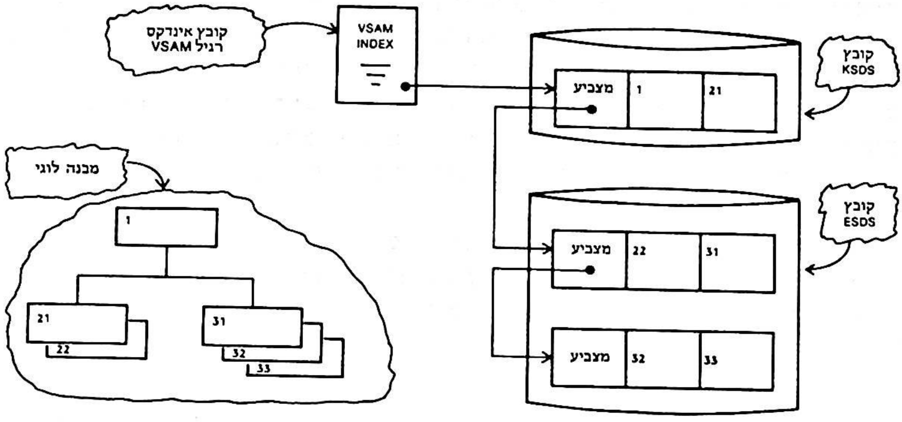
תרשים 9 – 11: מבנה קובץ HSAM

3.2 HISAM — Hierarchical Indexed Sequential Access Method

שיטה זו מאפשרת **גישה באמצעות אינדקס** לסגמנט השורש **וגישה סדרתית הירכית** לשאר הסגמנטים התלויים של הרשומה. המבנה ההירכי של הרשומה נשמר על ידי הקירבה הפיסית של הסגמנטים לפי הסדר ההירכי (מעלה—מטה, שמאל—ימין). הגישה לאחד מהסגמנטים התלויים מתבצעת על ידי סריקה סדרתית של כל הסגמנטים הקודמים לסגמנט המבוקש.

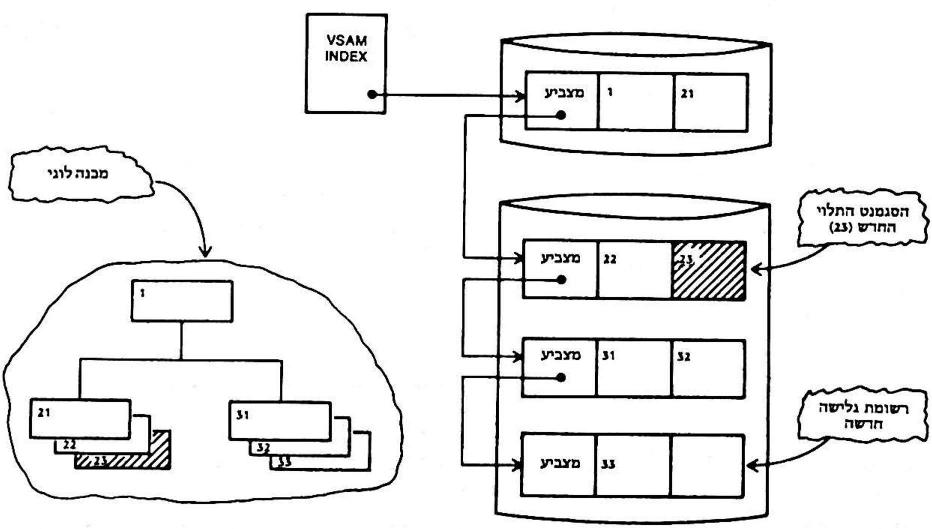
האחסון הפיסי מבוצע בקובץ VSAM רגיל בשיטת קובץ KSDS וקובץ ESDS. בשלב ראשון מוכנסים הסגמנטים לרשומות KSDS בעלות גודל קבוע. הרשומה הפיסית תכיל את סגמנט השורש ומספר סגמנטים תלויים נוספים, ככל שרשומת KSDS יכולה להכיל. אם יש צורך במקום אחסון לסגמנטים נוספים משתמשים ברשומות ESDS הקשורות על ידי מצביעים לרשומת KSDS. קובץ ESDS קרוי גם **שטח הגלישה** (Overflow Area).

תרשים 10—11 מדגים את מבנה קובץ HISAM. נשים לב לכך שהמבנה ההירכי נשמר על ידי הקירבה הפיסית של הסגמנטים בסדר הירכי וקישור רשומות פיסיות ביניהן בסדר עוקב.



תרשים 10 – 11: מבנה קובץ HISAM

הוספת סגמנט תלוי חדש לרשומת HISAM מתבצעת בצורה הבאה: מערכת DL/1 מאתרת את המקום שבו צריך להמצא הסגמנט החדש בהתאם לסדר ההירכי ומזיזה את כל הסגמנטים במקום אחד, כדי שיתפנה מקום עבור הסגמנט החדש. בגלל ההזזה יתכן שיהיה צורך לפתוח רשומת גלישה ESDS חדשה, אשר תכיל את הסגמנט האחרון בסדר ההירכי. הרשומה הפיסית החדשה מקושרת לשאר הרשומות באמצעות מצביע. בתרשים 11—11 מוצג מצב הקובץ לאחר הוספת סגמנט 23 במבנה ההירכי.



תרשים 11 – 11: הוספת סגמנט תלוי לרשומת HISAM

אם צריך להוסיף סגמנט שורש חדש, הוא יכתב מייד בשטח הגלישה ויקושר באמצעות מצביע לסגמנט השורש של הרשומה עם המפתח הגדול מהרשומה החדשה. בצורה זו אין צורך לבצע הזזות של רשומות בשטח הראשי של קובץ KSDS.

שיטת HISAM אינה תומכת בסגמנטים באורך משתנה. השיטה אינה מאפשרת קיום קשרים לוגיים בין בסיסי נתונים (הקשרים הלוגיים יוסברו בהמשך) ואינה מאפשרת להגדיר אינדקסים משניים. HISAM אינה משחררת שטח של רשומות שבוטלו. שטח זה משתחרר רק בזמן הארגון מחדש של הקובץ.

קיימת אפשרות להגדיר **מבנה אינדקס סדרתי פשוט** (Simple HISAM), עבור רשומות המכילות סגמנט שורש בלבד. שיטה זו משתמשת בקובץ KSDS בלבד ויוצרת למעשה קובץ אינדקס סדרתי רגיל. בכך היא מאפשרת לגשת לקובץ אינדקס סדרתי רגיל מתוך תוכניות DL/1.

3.3 HDAM — Hierarchical Direct Access Method

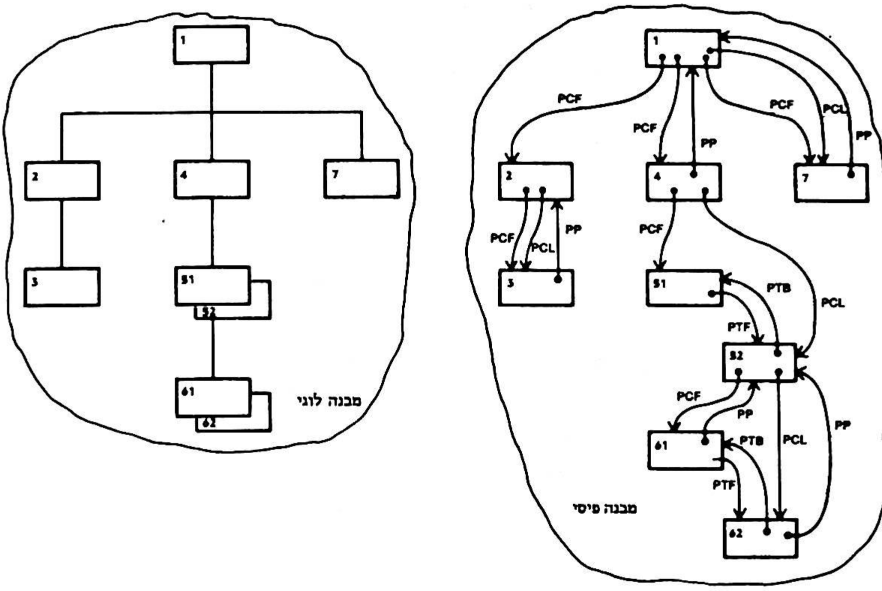
שיטת הגישה HDAM מאפשרת גישה ישירה לסגמנט השורש על פי כתובתו. הכתובת בדיסק נקבעת על פי המפתח ושיטת הרנדומיזציה של מערכת DL/1. שאר הסגמנטים קשורים ביניהם באמצעות מצביעים. השיטה מיושמת בקובץ ESDS של VSAM.

שיטת HDAM מאפשרת את הגישה המהירה ביותר לרשומות. היא גם מאפשרת ניצול יעיל של שטח האחסון, מכיון שבזמן ביטול סגמנט משתחרר מיד השטח שהתפנה ומנוצל לאחסון סגמנטים חדשים. שיטת HDAM מאפשרת שימוש באינדקסים משניים ובקשרים לוגיים בין בסיסי נתונים פיסיים שונים.

מערכת DL/1 הופכת את מבנה העץ הכללי למבנה של עץ בינרי שבו כל סגמנט מצביע על סגמנט הבן שלו ועל הסגמנט התאום שלו. כל הסגמנטים התאומים קשורים ביניהם **בשרשרת תאומים** (Twin Chain) אשר אפשר להגדיר אותה בכיוון לפנים או לאחור. המערכת משתמשת במצביעים שונים לשם אחסון המבנה ההירכי בשיטת HDAM:

— PCF (Physical Child First): מצביע על סגמנט הבן הראשון,
— PCL (Physical Child Last): מצביע על סגמנט הבן האחרון,
— PTF (Physical Twin Forward): מצביע על סגמנט התאום הבא,
— PTB (Physical Twin Backward): מצביע על סגמנט התאום הקודם,
— PP (Physical Parent): מצביע מסגמנט בן לסגמנט האב שלו.

מנהל בסיס הנתונים יכול לבחור בין שתי שיטות של ניהול המצביעים: **מצביעים הירכיים** (Hierarchic pointers), **ומצביעים לסגמנט בן ותאום** (Child/twin pointers). בשיטה של מצביעים הירכיים לכל סגמנט יש מצביע אחד לסגמנט הבא בסדר ההירכי של הרשומה, למעט הסגמנט האחרון. בשיטה השניה לכל סגמנט יש מצביעים אחדים לסגמנט בן, כמספר הסגמנטים מסוג בן. לכל סגמנט בן יש מצביעים לסגמנטים התאומים ולסגמנטים הבנים שלו. בשיטה זו המצביעים PCF ו־PTF הם הכרחיים בעוד ששאר המצביעים ניתנים לבחירה. כל המצביעים נשמרים בחלק קבוע בתחילת הרשומה והם שקופים מבחינת המשתמש.

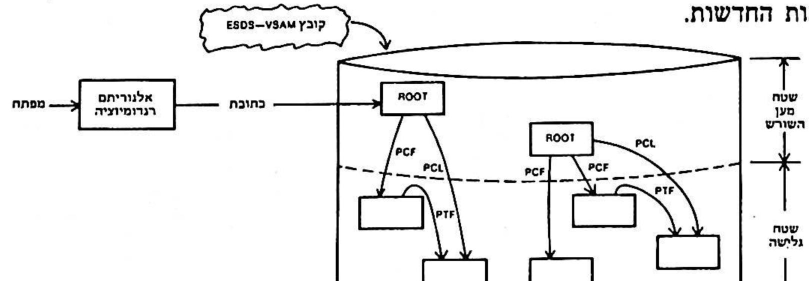


תרשים 12 – 11: הפיכת עץ כללי לעץ בינרי עם מצביעים ב־HDAM

תרשים 12—11 מציג מבנה HDAM המשתמש בשיטת מצביע בן/תאום. כדי לאחסן את המבנה הזה משתמש DL/1 בשני שטחים נפרדים בקובץ ESDS—VSAM:

— **שטח מען השורש** (RAA — Root Addressable Area) מיועד לאחסנת סגמנט השורש ומספר סגמנטים תלויים נוספים בהתאם לגודל הרשומה הפיסית שהוגדר. כתובת הרשומה נקבעת על ידי הפעלת תהליך רנדומיזציה על המפתח שלה. לכל גוש בשטח זה יש כתובת, ותהליך הרנדומיזציה מתרגם את מפתח הרשומה לכתובת של גוש. אם מספר מפתחות שונים מייצרים את אותה הכתובת נבנית **שרשרת של הפניות הזהות** (Synonyms Chain).

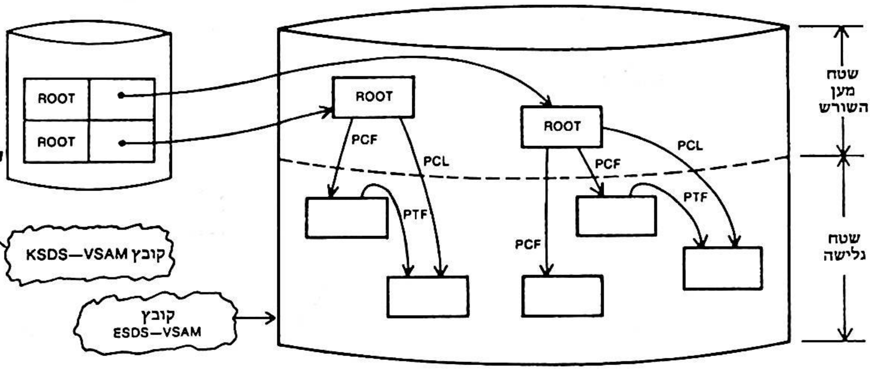
— **שטח גלישה** (Overflow Area) משמש לאחסון כל הסגמנטים התלויים של הרשומה שלא היה להם מקום ברשומה שבשטח מען השורש. אם אלגוריתם הרנדומיזציה מייצר לשתים או יותר רשומות את אותה הכתובת בשטח מען השורש מתבצע חיפוש לינארי לפנים עד לאיתור מקום פנוי, ומתבצע קישור באמצעות מצביעים בין הרשומה בכתובת המקורית לבין הרשומות החדשות.



תרשים 13 – 11: מבנה קובץ HDAM

HIDAM — Hierarchical Indexed Direct Access Method 3.4

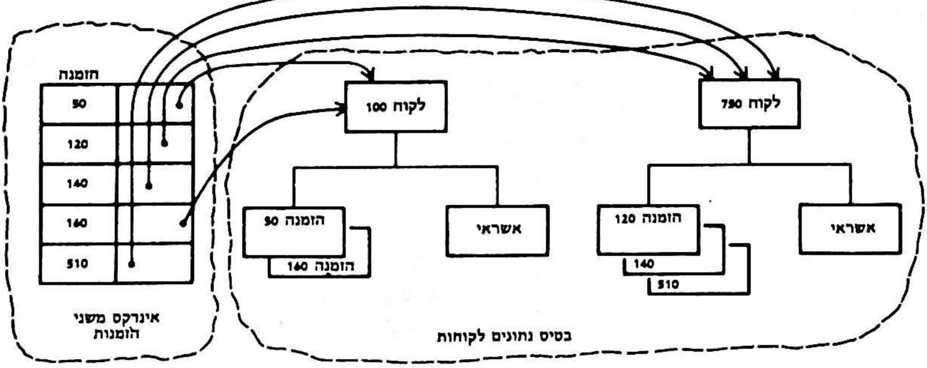
שיטת הגישה HIDAM דומה לשיטת הגישה HDAM. בנוסף לקובץ ESDS המכיל את הרשומות משתמשת מערכת DL/1 גם בקובץ KSDS אשר מכיל את המפתח של הרשומה ומצביע לכתובתה. קובץ האינדקס KSDS מאפשר עיבוד סדרתי של הקובץ.



תרשים 14 – 11: מבנה קובץ HIDAM

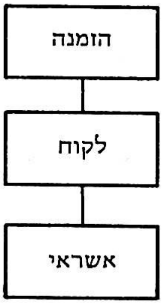
3.5 אינדקס משני (Secondary Index)

בכדי לאפשר גישה מהירה לרשומה בעזרת מפתחות נוספים מנהלת מערכת DL/1 אינדקסים משניים. שדה המפתח הנוסף יכול להמצא בסגמנט השורש עצמו או בכל סגמנט תלוי אחר ברשומה. תפקיד חשוב של האינדקס המשני הוא לאפשר גישה ישירה לסגמנט ברמה נמוכה יותר מרמת השורש. הטיפול באינדקס המשני נעשה על־ידי המערכת ולכן הוא "שקוף" מבחינת המשתמש.



תרשים 15 – 11: שימוש באינדקס משני

תרשים 15—11 מציג דוגמא של בניית אינדקס משני בבסיס נתונים לקוחות. המפתח בסגמנט השורש הוא מספר לקוח, אך אם רוצים לגשת לרשומה גם לפי מספר ההזמנה של הלקוח יש לבנות אינדקס משני. השימוש באינדקס המשני מציג בפני המשתמש מבנה לוגי חדש שבו סגמנט ההזמנה הופך להיות סגמנט השורש.



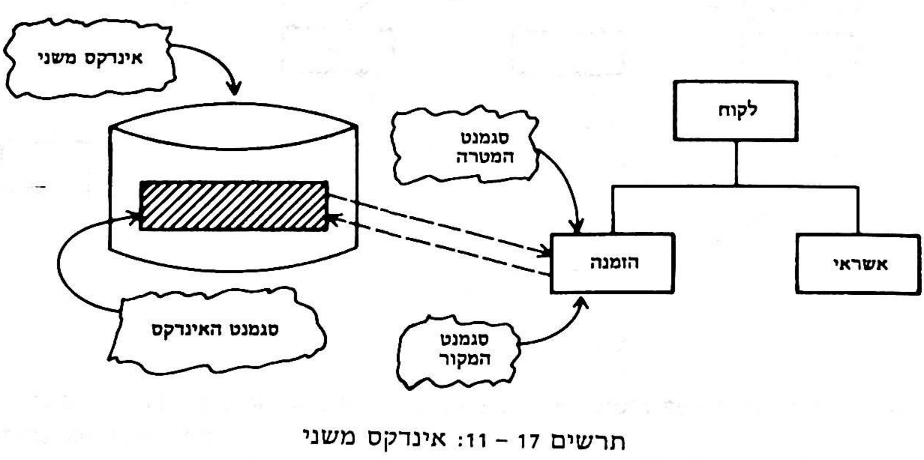
תרשים 16 - 11: מבנה לוגי חדש כתוצאה משימוש באינדקס משני

כדי לבנות את האינדקס המשני על המשתמש להגדיר את הסגמנטים הבאים:

— **סגמנט המקור** (Index Source Segment): זהו הסגמנט המכיל את שדה המפתח שלפיו יש לבנות את האינדקס המשני. האינדקס המשני שיבנה יהיה ממויין לפי מפתח זה. בדוגמא הקודמת זהו סגמנט ההזמנה.

— **סגמנט המטרה** (Index Target Segment): זהו הסגמנט שהאינדקס המשני מצביע עליו. סגמנט המטרה מציין איזה סגמנט ישלף כאשר מעבדים את הקובץ באמצעות האינדקס המשני. סגמנט המטרה וסגמנט המקור יכולים להיות זהים, אך אין זו חובה. בתרשים 15—11 סגמנט המטרה של "אינדקס משני הזמנות" הוא סגמנט השורש "לקוח". בתרשים 17—11 סגמנט המטרה שווה לסגמנט המקור.

— **סגמנט האינדקס** (Index Pointer Segment): סגמנט זה נמצא בתוך קובץ האינדקס המשני והוא מנוהל על ידי DL/1. כל סגמנט אינדקס מצביע לאינדקס מטרה אחד. המפתח של סגמנט האינדקס הוא המפתח המשני שנלקח מתוך סגמנט המקור.



תרשים 17 - 11: אינדקס משני

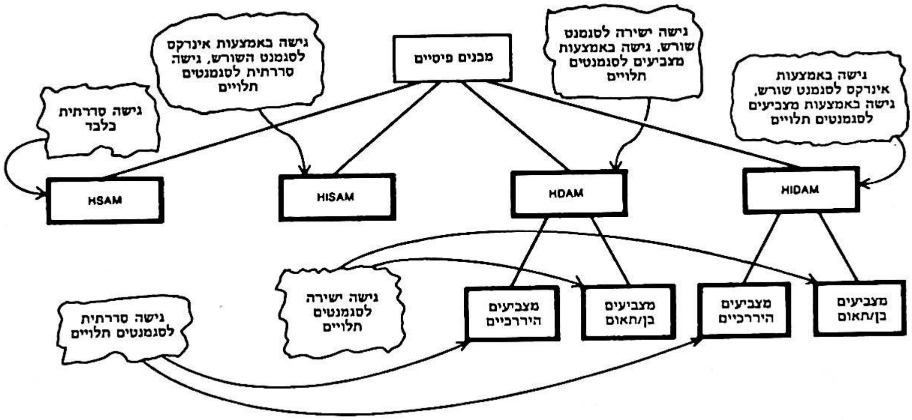
3.6 השוואה בין המבנים הפיסיים

ארבעה מבנים פיסיים או שיטות גישה שונים מופעלים על ידי IMS: HSAM, HISAM, HIDAM, HDAM. הם נבדלים זה מזה בשלשת המאפיינים העיקריים:

— **מהירות איתור (חיפוש):** לאיתור רשומה בשיטת HSAM יש לסרוק כמחצית מבסיס הנתונים ולכן אין זה מבנה מתאים עבור גישות ישירות. שיטת HDAM היא המהירה ביותר לאיתור סגמנט השורש מכיון שהיא משתמשת בתהליך חישובי ולא בסריקת אינדקסים. בשיטת HISAM רשומת בסיס הנתונים מאוחסנת בדרך כלל בצורה רציפה ללא מצביעים מסוג כלשהו, למעט מצביעים הקושרים את הרשומה לסגמנטים הנמצאים בשטח גלישה. שיטה זו יעילה מאד כאשר יש לסרוק את המבנה ההירככי בסדר מעלה־מטה־שמאל־ימין, כי אין צורך לבצע גישות פיסיות רבות לקבלת כל הסגמנטים.

— **הוספת רשומה:** במבנה HSAM אין אפשרות להוסיף רשומות או סגמנטים חדשים. במבנה HDAM הוספת רשומה או סגמנט היא המהירה ביותר. במבנה HIDAM מהירות הוספה גבוהה אך יש צורך גם לעדכן את האינדקס. במבנה HISAM יש לבצע הזזה פיסית של כל הסגמנטים כדי לשמור על הסדר ההירככי שלהם ולכן מהירות ההוספה איטית יותר.

— **ניצול שטח אחסנה:** במבנה HSAM הניצול הוא הטוב ביותר כי אין תקורה הנובעת משימוש במצביעים. מבנה HISAM זקוק למקום עבור האינדקס אבל אינו משתמש במצביעים לקישור הסגמנטים לשטח גלישה, למעט מצביעים מרשומת השורש לשטח הגלישה. מבנה HISAM אינו משחרר את השטחים שמתפנים לאחר ביטול סגמנטים. מבנה HDAM אינו זקוק למקום עבור אינדקסים אבל משתמש במגוון מצביעים לקישור הסגמנטים למבנה הירככי. HDAM משחרר את השטח שמתפנה בעקבות ביטול סגמנט ולכן ניצול שטח האחסון טוב יותר מאשר ב־HISAM. מבנה HIDAM זקוק למקום עבור האינדקס ועבור המצביעים.



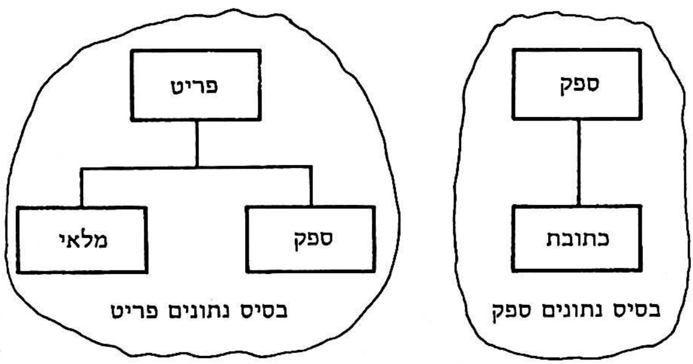
תרשים 18 -- 11: מבנים פיסיים ב־IMS

תרשים 18—11 מסכם את ארבעת המבנים הפיסיים או שיטות הגישה הקיימים ב־IMS ומדגיש את המאפיינים העיקריים של כל אחד מהם.

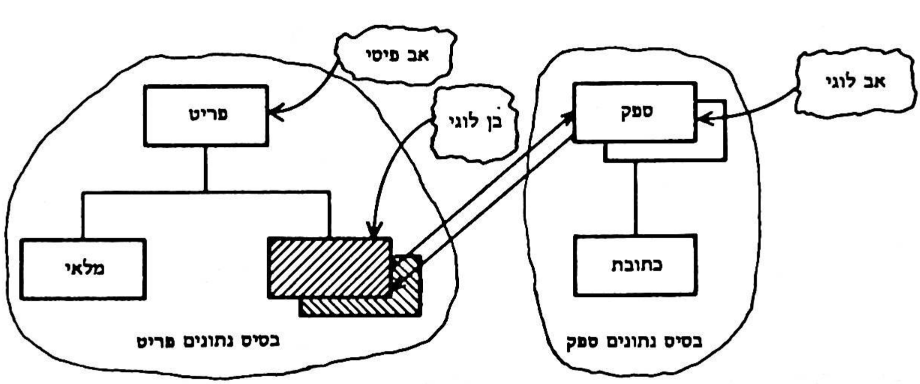
4. קשרים לוגיים (Logical Relationship)

ב־IMS אפשר לקשור סגמנטים המופיעים בשני בסיסי נתונים פיסיים או יותר, או לקשור סגמנטים מאותו בסיס נתונים פיסי בצורה שונה מהמבנה ההירככי הפיסי. הקשרים הלוגיים מאפשרים קבלת מבנים לוגיים נוספים למבנה הלוגי המתקבל מתוך בסיס הנתונים הפיסי. בצורה זו מוגדרות למעשה הירככיות חדשות המאפשרות גישה נוחה ושונה אל הנתונים. הרשומה החדשה שנוצרת הינה **רשומה לוגית** (Logical Record) להבדיל מהרשומה הפיסית שהכרנו עד כה.

הצורך בקשרים לוגיים מתעורר כאשר קיימים נתונים זהים במבנים הירככיים שונים. תרשים 19—11 מציג דוגמא לשני בסיסי נתונים פיסיים, בסיס נתונים "פריט" ובסיס נתונים "ספק". בבסיס נתונים פריט יש לנהל נתונים אודות הספקים של כל פריט בעוד שבבסיס נתונים ספק יש לנהל נתונים אודות כל הספקים המוכרים במערכת. זוהי דוגמא לכפילות נתונים.



תרשים 19 – 11: כפילות נתונים בשני בסיסי נתונים פיסיים



תרשים 20 – 11: שימוש בקשרים לוגיים למניעת כפילות נתונים

ב־IMS אפשר למנוע את כפילות הנתונים על־ידי שימוש בקשרים לוגיים. ניתן להחליף את סגמנט הספק בבסיס נתונים פריט במצביע אל רשומת הספק המתאים בבסיס נתונים ספק. במקרה זה נאמר שסגמנט השורש של רשומת הספק הוא **סגמנט אב לוגי** (Logical Parent) של סגמנט הספק בבסיס נתונים פריט. סגמנט הספק בבסיס נתונים פריט יקרא **סגמנט בן לוגי** (Logical Child). בדוגמא שבתרשים 20—11 יתכן שאותו פריט מסופק על ידי ספקים שונים. במקרה זה יהיו מספר סגמנטים של מצביעים מבסיס נתונים פריט לבסיס נתונים ספק.

לסגמנט הבן הלוגי יש שני אבות, אב פיסי ואב לוגי. לסגמנט הבן הלוגי ניתן לגשת מסגמנט האב הפיסי דרך המסלול הפיסי, ואפשר לגשת אליו מסגמנט האב הלוגי דרך המסלול הלוגי. כאשר סגמנט בן לוגי מוצג למשתמש הוא מכיל את **המפתח המשורשר של האב הלוגי** (LPCK — Logical Parent Concatenated Key) ואחריו את הנתונים בסגמנט עצמו. מבנה סגמנט בן לוגי מוצג בתרשים 21—11. המפתח המשורשר (LPCK) מוצג למשתמש תמיד כאשר הוא ניגש לסגמנט דרך המסלול הפיסי. עם הוספת סגמנט בן לוגי חדש לבסיס הנתונים יוצג המפתח המשורשר שלו, כדי שמערכת DL/1 תוכל לבצע את הקישור הלוגי המתאים.

Intersection Data נתונים המופיעים בסגמנט בן לוגי ואב לוגי	LPCK	Prefix מצביעים

תרשים 21 - 11: מבנה סגמנט בן לוגי

בדוגמא שבתרשים 20—11, בעת הוספת סגמנט ספק חדש לבסיס נתונים פריט יכיל סגמנט הבן הלוגי את מספר הספק ונתונים נוספים הנוגעים לפריט ולספק, כגון תאריך תחילת אספקה. בעת קריאת סגמנט בן לוגי יוצגו למשתמש נתונים אשר נמצאים בסגמנט עצמו ונתונים נוספים אשר נמצאים ברשומת האב הלוגי. בדוגמא זו יבוצע הקישור מרשומת הפריט לרשומת הספק בבסיס נתונים ספק באמצעות השדה "מספר ספק".

הצגת קשרים לוגיים מחייבת שימוש במצביעים מיוחדים בנוסף למצביעים הנהוגים לתיאור הקשרים הפיסיים. מערכת IMS משתמשת בקשרים הלוגיים הבאים, אשר מקבילים לסוגי הקשרים הפיסיים:

— LCF (Logical Child First): מצביע על הבן הלוגי הראשון,
— LCL (Logical Child Last): מצביע על הבן הלוגי האחרון,
— LTF (Logical Twin Forward): מצביע על הסגמנט התאום הלוגי הבא,
— LTB (Logical Twin Backward): מצביע על הסגמנט התאום הלוגי הקודם,
— LP (Logical Parent): מצביע מסגמנט בן לוגי לסגמנט אב לוגי.

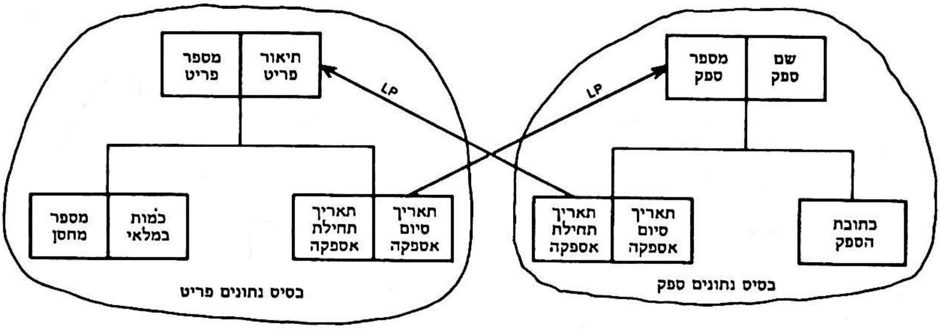
4.1 שיטות להצגת מבנים לוגיים

קשרים לוגיים אפשר ליישם בשלש שיטות:

4.1.1 קשר לוגי חד־כיווני

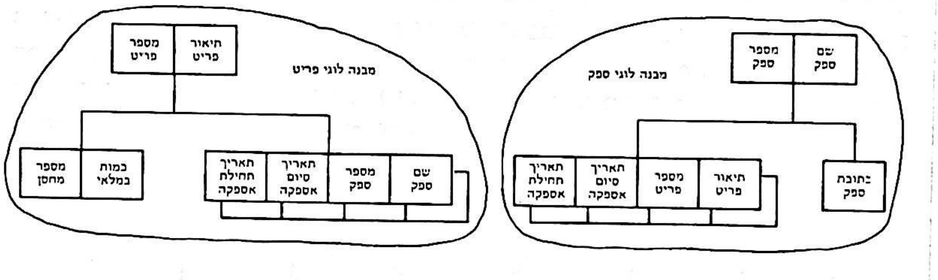
בשיטת קשר לוגי חד־כיווני (Monodirectional Logical Relationship) משתמשים במצביע LP אשר מורה מסגמנט הבן הלוגי אל סגמנט האב הלוגי. נניח שקיים קשר רב—רב־

ערכי בין פריטים לספקים. כדי לנהל קשר זה ניתן להשתמש בקשר לוגי חד־כיווני בין שני המבנים ההיררכיים.



תרשים 22 – 11: קשר לוגי חד־כיווני

הקשר הלוגי החד־כיווני בין הסגמנטים, כפי שמופיע בתרשים 22—11, מאפשר הצגת שני מבנים לוגיים חדשים בנוסף למבנים הלוגיים המבוססים על המבנה הפיסי של בסיסי הנתונים. שני מבנים לוגיים חדשים אלו מוצגים בתרשים 23—11. כפי שניתן לראות, מבנה לוגי פריט מציג בפני המשתמש את נתוני הפריט יחד עם נתוני הספק (או ספקים) של הפריט. מבנה לוגי ספק מציג בפני המשתמש את נתוני הספק יחד עם נתוני הפריטים שהוא מספק. נשים לב לעובדה שהשימוש בקשר לוגי חד־כיווני אינו מונע כפילות נתונים ומשאיר את הטיפול בהם באחריות המשתמש.

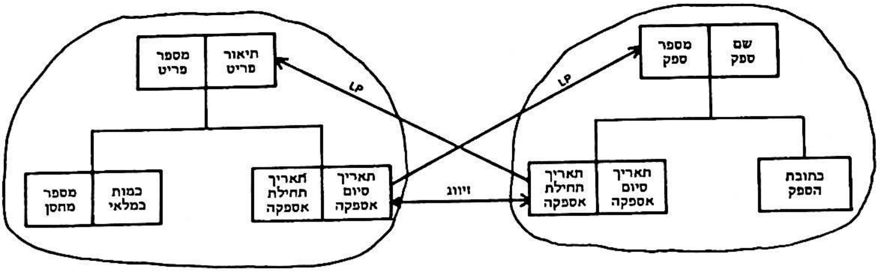


תרשים 23 – 11: מבנים לוגיים המבוססים על קשר לוגי חד־כיווני

4.1.2 זיווג פיסי בקשר לוגי דו־כיווני

מערכת IMS משתמשת בשיטה זו (Physically Paired Bidirectional Logical Relationship) לשם קישור ישיר בין שני הסגמנטים המופיעים בשני המבנים ההיררכיים. בשיטה הקודמת ניהול הנתונים הכפולים בשני הסגמנטים הוא באחריות המשתמש, וכאן מעדכנת מערכת DL/1 באופן אוטומטי את הסגמנט השני בעת עדכון הסגמנט הראשון, ולהיפך. לדוגמא, עדכון שדה "תאריך תחילת אספקה" בהירכיה של הפריט יגרום לעדכון

מקביל של שדה זה בהיררכיה של הספק ולהיפך. המבנים הלוגיים המתקבלים בשיטה זו זהים לאלו המתקבלים בשיטה הקודמת.

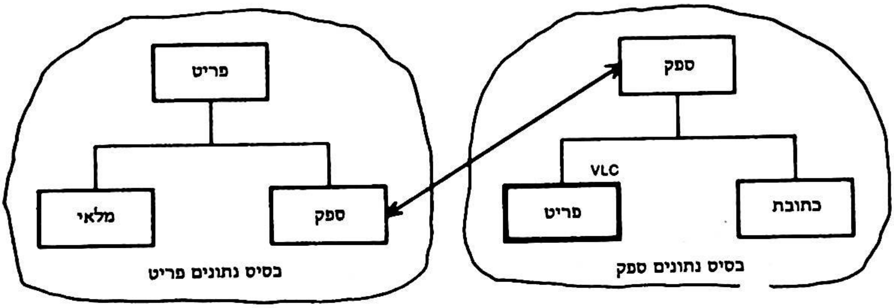


תרשים 24 – 11: זיווג פיסי בקשר לוגי דו־כיווני

4.1.3 זיווג בפועל בקשר לוגי דו־כיווני

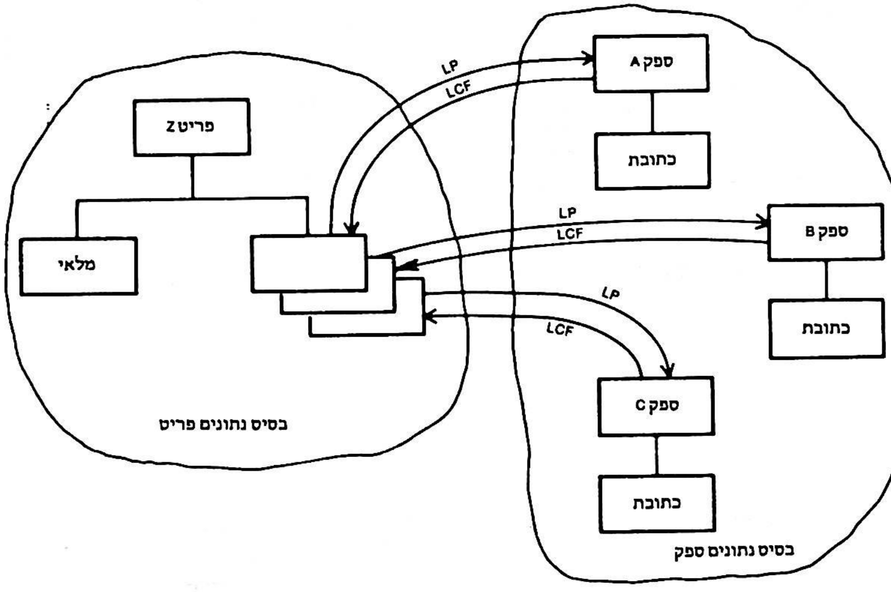
כדי למנוע את כפילות הנתונים אשר קיימת בשתי השיטות הקודמות פותחה שיטת קישור נוספת (Virtually Paired Bidirectional Logical Relationship). בשיטה זו רק אחד משני הסגמנטים מופיע בצורה פיסית בבסיס הנתונים ואילו הסגמנט השני מופיע בפועל בלבד.

להגדרת הקשר בין האב הלוגי לבין הבן הלוגי משתמשת מערכת DL/1 **בסגמנט בן בפועל** (VLC — Virtual Logical Child). הוא מוגדר כסגמנט תלוי של האב הלוגי ואינו קיים באופן פיסי ביחידת האחסון. מטרתו היחידה היא להגדיר את נקודת המבט של האב הלוגי על הנתונים המופיעים בסגמנט הבן הלוגי. באמצעות סגמנט הבן בפועל מתאפשרת הגישה מסגמנט האב הלוגי לסגמנט הבן הלוגי. סגמנט זה מאפשר גם להגדיר את סדר המיון של הבנים הלוגיים במסגרת קשר התאומים. מכיון שניתן לגשת לבן הלוגי הן מהאב הפיסי והן מהאב הלוגי, מתקבל קשר לוגי דו־כיווני (תרשים 25—11).



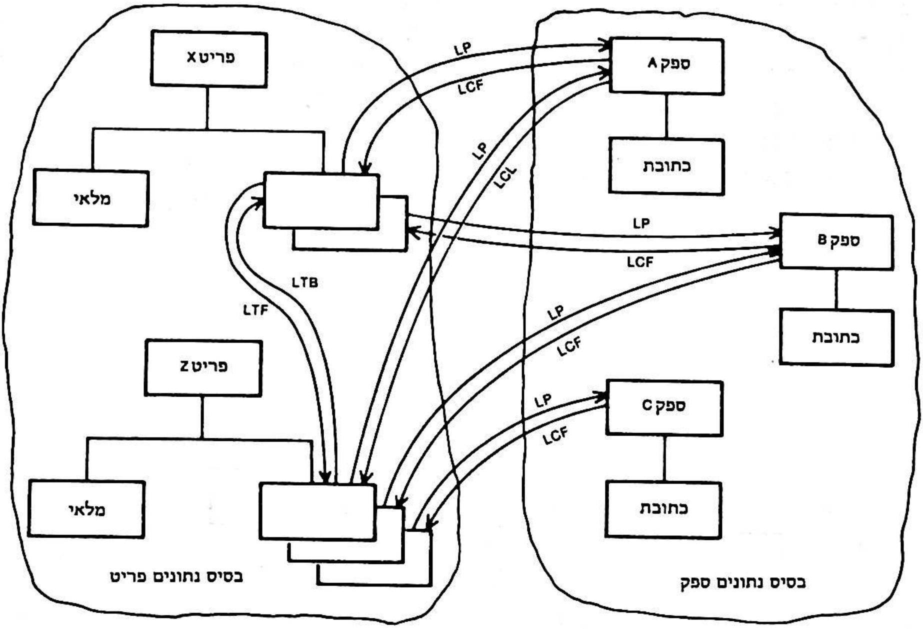
תרשים 25 – 11: קשר לוגי דו־כיווני

הקשר הלוגי הדו־כיווני בין שני בסיסי הנתונים הפיסיים מאפשר הגדרת מבנים לוגיים חדשים אשר מכילים סגמנטים משני בסיסי הנתונים הפיסיים. תוכנית היישום פועלת על המבנה הלוגי כאילו הוא המבנה הפיסי של הנתונים. עבור כל פריט היא יכולה לקבל את כל הספקים והכתובות שלהם, ולהיפך — עבור כל ספק ניתן לקבל את כל הפריטים שהוא מספק ואת המלאי הנוכחי שלהם. באופן פיסי אף אחד מהמבנים האלה אינו קיים, אבל ניתן לפעול עליהם כאילו היו קיימים. תוכנית היישום אינה מטפלת במצביעים והיא רואה את המבנה הלוגי העומד לרשותה כמבנה היררכי פשוט.

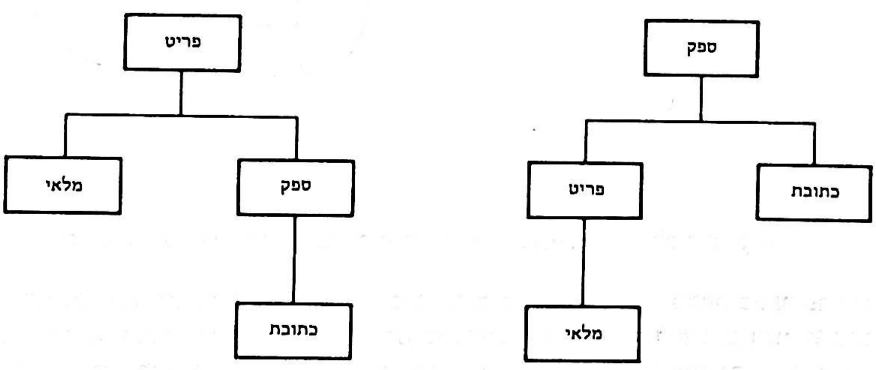


תרשים 26 - 11: קשר לוגי דו־כיווני בין רשומת פריט לבין ספקים

תרשים 27—11 מציג דוגמא של הקשרים הלוגיים כאשר ספק אחד מספק מספר פריטים שונים. ספק A מספק פריטים X, Z ולכן יש מצביעים (LP) מהם אל סגמנט השורש שלו. מסגמנט השורש ספק ישנו מצביע לפריט הראשון (LCF) ולפריט האחרון (LCL). בין הסגמנטים בשתי רשומות הפריט קיימים קשרים של תאומים לוגיים (LTB ,LTF). המבנים לוגיים המתקבלים מקשרים לוגיים דו־כיווניים מוצגים בתרשים 28—11.



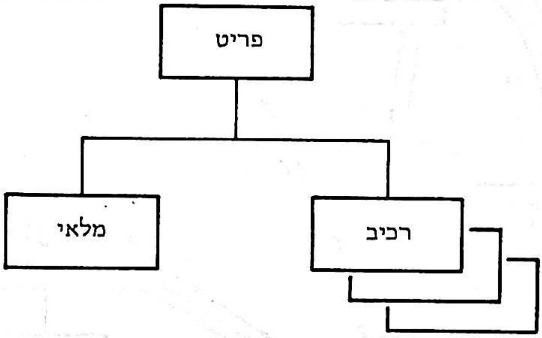
תרשים 27 – 11: קשר לוגי דו כיווני בין רשומות פריט לבין רשומות ספק



תרשים 28 – 11: מבנים לוגיים המתקבלים מקשרים לוגיים דו־כיווניים

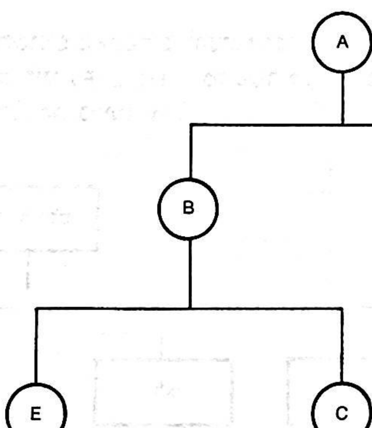
4.2 הצגת מבנה עץ מוצר כמבנה לוגי

אפשר להשתמש בקשרים לוגיים גם עבור סגמנטים אשר נמצאים באותו מבנה פיסי. נדגים זאת באמצעות מבנה עץ מוצר של פריט. הפריט יכול להיות הרכבה סופית, תת־הרכבה, פריט בסיסי מעובד או חומר גלם. שמירת הנתונים על הרכיבים בתוך הסגמנטים התלויים היתה גורמת לכפילות נתונים רבה מכיון שאותו רכיב (תת־הרכבה או חומר גלם) יכול להופיע במספר רב של פריטים סופיים.



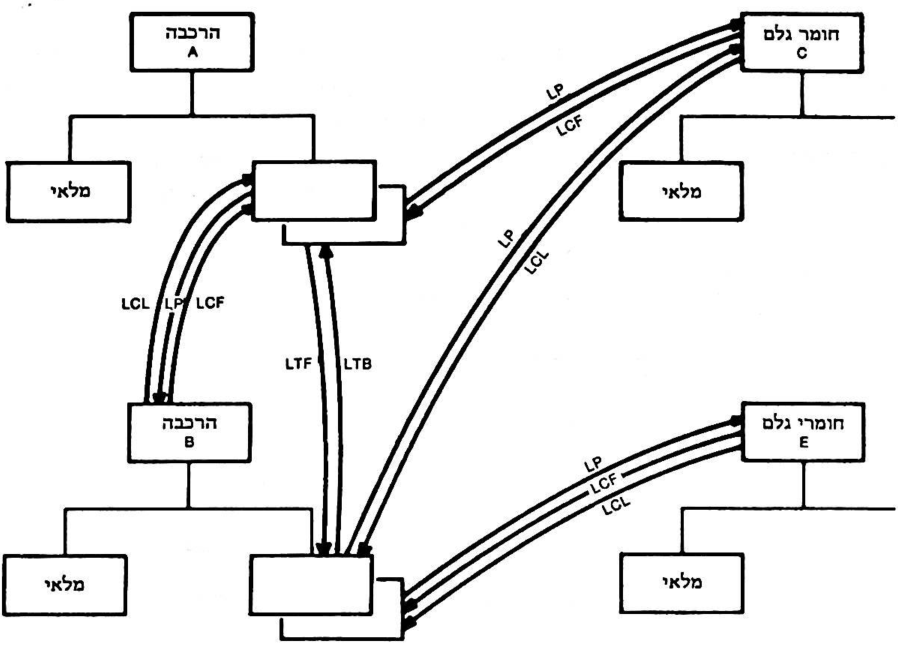
תרשים 29 – 11: מבנה רשומת פריט - עץ מוצר

למניעת הכפילות בנתונים ניתן להשתמש בקשרים לוגיים. עץ המוצר של פריט A מוצג בתרשים 30—11. הפריטים A ו־B הם הרכבות והפריטים E וְ־C הם חומרי גלם או פריטים בסיסיים מעובדים.



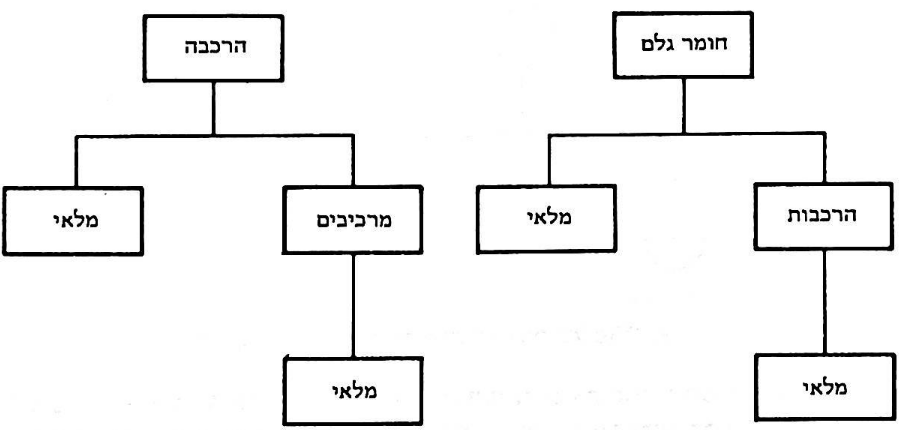
תרשים 30 – 11: דוגמא לעץ מוצר של פריט A

תרשים 31—11 מציג את המצביעים השונים הקושרים את סגמנט השורש של הפריטים השונים עם סגמנטים תלויים של פריטים אחרים. בצורה זו מתארים מבנה עץ מוצר ללא כפילות נתונים.



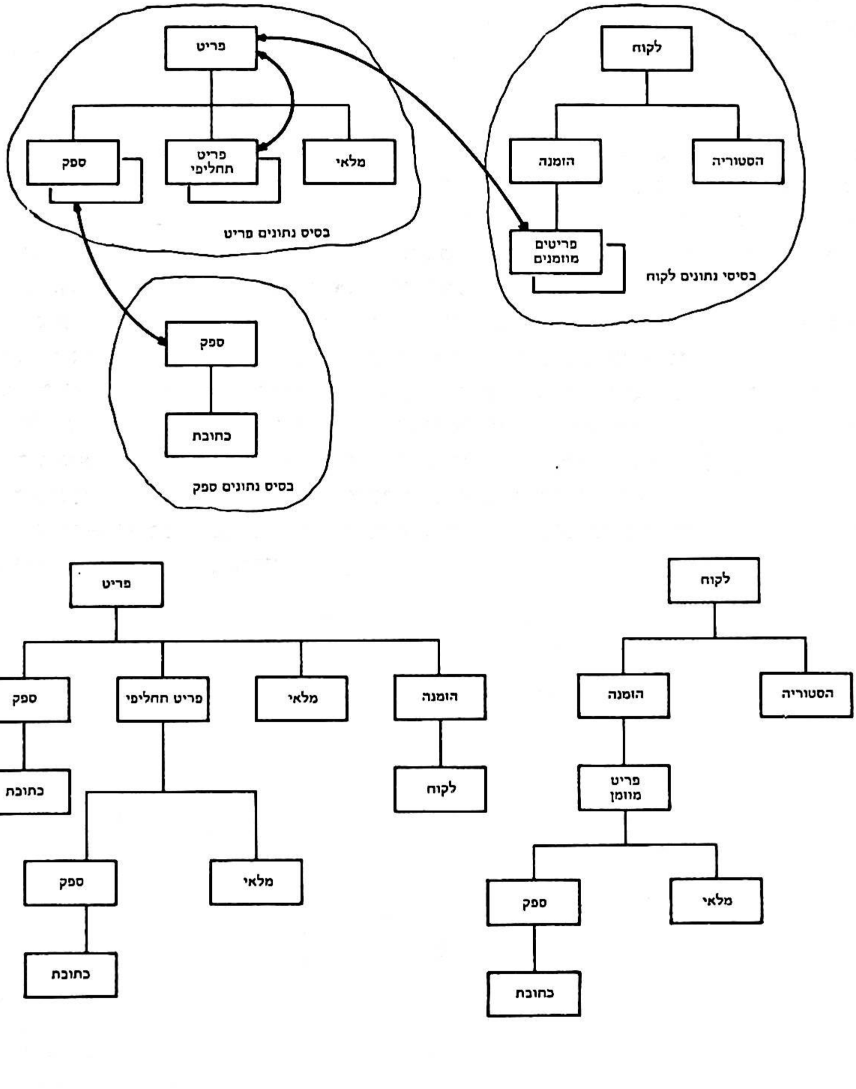
תרשים 31 – 11: מצביעים הדרושים ליישום עץ המוצר

הקשרים הלוגיים השונים מאפשרים להציג מבנה לוגי חדש שבו **אין** נתונים כפולים כלשהם. השימוש בקשר תאום לוגי (LTB, LTF) מאפשר לקבל מבנה המכיל את כל ההרכבות שבהן משתתף חומר גלם כלשהו, כלומר **אילו רשומות כלולות באילו רשומות או היכן בשימוש** (Where-in-Use).



תרשים 32 – 11: מבנים לוגיים המתקבלים משימוש בקשרים לוגיים

השימוש בקשרים הלוגיים מאפשר בניית מבנים לוגיים מורכבים ביותר אשר מכילים סגמנטים ממספר בסיסי נתונים פיסיים. תרשים 33—11 מציג דוגמא של קשרים לוגיים בין שני בסיסי נתונים פיסיים ובין סגמנטים באותו בסיס נתונים פיסי, ושל שני המבנים הלוגיים הנובעים מאותם קשרים.



תרשים 33 – 11: קשרים לוגיים ומבנים לוגיים

4.3 בסיס נתונים לוגי

בסיס נתונים לוגי (Logical Data Base) הינו אוסף של סגמנטים המסודרים במבנה הירכי בדיוק כמו בסיס נתונים פיסי. מקור הסגמנטים בבסיס נתונים לוגי הוא בסיס נתונים פיסי אחד או בסיסי נתונים פיסיים אחדים. סגמנט כלשהו בבסיס הנתונים הלוגי יכול להכיל את נתוני הסגמנט בלבד או גם נתונים נוספים שמופיעים בסגמנט האב הפיסי או בסגמנט האב הלוגי.

בבסיס נתונים לוגי, סגמנט B יכול להיות בן של סגמנט A אם קיים מצביע כלשהו מסגמנט A אל סגמנט B. מצביע זה יכול להיות המצביע הרגיל מסגמנט אב פיסי לסגמנט בן פיסי, או מצביע מסגמנט בן לוגי או סגמנט אב לוגי. ניתן לומר שבבסיס נתונים לוגי הקשרים אב־בן יכולים להיות קשרי בן־אב באחד מבסיסי הנתונים הפיסיים.

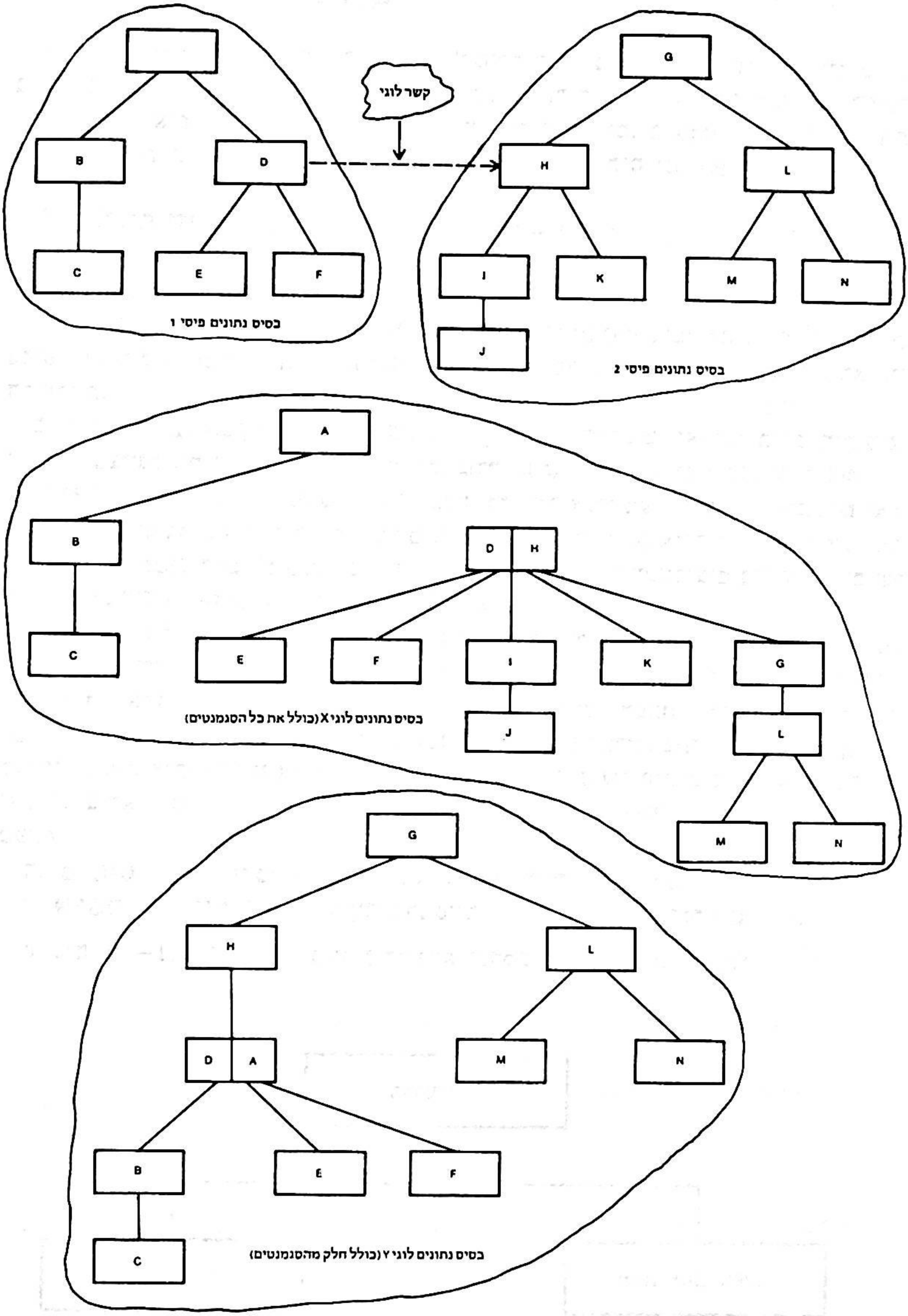
בעת בניית בסיס נתונים לוגי יש לקיים שני כללים:

— **סגמנט השורש** בבסיס נתונים לוגי **חייב** להיות סגמנט שורש בבסיס נתונים פיסי כלשהו.

— כאשר **הקשר אב־בן** בבסיס הנתונים הלוגי הוא קשר אב־בן בבסיס נתונים פיסי, **לא** ניתן להשתמש באב הפיסי שנית בבסיס הנתונים הלוגי. כלומר, אם סגמנט A הוא אב של סגמנט B גם בבסיס נתונים לוגי וגם בבסיס נתונים פיסי, סגמנט A לא יוכל להיות בן של B בבסיס נתונים לוגי גם אם קיים בסגמנט B מצביע אל סגמנט האב A.

החשיבות של בסיסי נתונים לוגיים נובעת מהעובדה שניתן לנצל אותם לייצוג קשרים רב—רב־ערכיים בין שתי יישויות. כל מבנה רשתי ניתן לייצוג באמצעות אוסף של בסיסי נתונים לוגיים עם ירידה קטנה יחסית בזמן התגובה לאחזור. עם זאת נראה ששיטת הטיפול ברשתות לפי המלצות CODASYL הינה ישירה ופשוטה יותר. למרות הקשרים הלוגיים המבנה המוצג למשתמש הוא תמיד מבנה הירכי. בזמן פעולות עדכון עליו להכיר את המבנה ההירכי של בסיס הנתונים הפיסי, מכיון שקיימות מגבלות מסויימות לעדכון בבסיס נתונים לוגי.

תרשים 11—34 מציג שני בסיסי נתונים פיסיים ושני בסיסי נתונים לוגיים אשר מופקים באמצעות הקשר הלוגי שביניהם.



תרשים 34 – 11: בסיסי נתונים פיסיים ובסיסי נתונים לוגיים המופקים מהם

5. שפה להגדרת נתונים

מערכת ניהול בסיסי נתונים כוללת שפה להגדרת הנתונים אשר באמצעותה מגדיר מנהל בסיס הנתונים את המבנה הפיסי של בסיסי הנתונים ואת הקשרים הלוגיים שביניהם. השפה הינה למעשה אוסף של פקודות מקרו עם מגוון של פרמטרים. ב־IMS קיימות שתי שפות להגדרת נתונים: שפה להגדרת סכימה (DBD) ושפה להגדרת תת־סכימה (PSB).

5.1 הגדרת סכימה (DBD — Data Base Definition)

הפקודות המשמשות להגדרת הסכימה:

א. **DBD**: הפקודה הראשונה קובעת את שם בסיס הנתונים ואת שיטת הארגון שלו. להגדרת בסיס נתונים לוגי יש להשתמש בפרמטר LOGICAL במקום שם של אחת משיטות הארגון הבסיסיות.

ב. **DATASET**: הפקודה מתארת את הקובץ הפיסי המשמש לאחסון בסיס הנתונים. הפקודה מציינת את שם הקובץ, סוג הדיסק, גודל הגוש ומספר תכונות פיסיות נוספת.

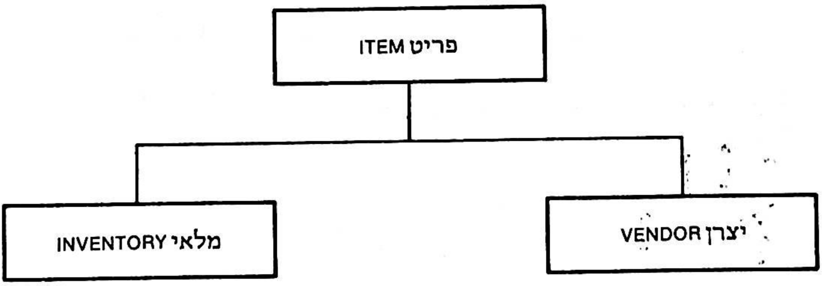
ג. **SEGM**: להגדרת הסגמנטים יש להשתמש. בפקודה אשר מציינת את שם הסגמנט ואת שם סגמנט האב במבנה ההיררכי. פרמטרים נוספים של הפקודה הם אורך הסגמנט בבתים, סוג המצביעים מסגמנט האב לסגמנט הבן (PCF, PCF, PCL) וסוג המצביעים בין התאומים של אותו סגמנט (PTF , PTF, PTB, או סגמנט ללא תאומים).

ד. **FIELD**: כל שדה בסגמנט יש להגדיר באמצעות פקודה אשר מכילה כפרמטרים את שם השדה, ציון שדה מפתח, גודל השדה בבתים וסוג השדה (נומרי, אלפא, נקודה צפה וכדומה). רק שדה אחד בסגמנט יכול להיות מוגדר כשדה מפתח. לדוגמא, הפרמטר NAME=(ITEM-NO, SEQ, U) מגדיר את השדה ITEM-NO כשדה מפתח. האות U (Unique) מציינת שתחת אותו סגמנט אב לא יתכנו שני מופעים שונים של סגמנט בן עם אותו מפתח, ואם תכתב האות M (Multiple) היא תציין שיתכנו מספר מופעים של סגמנט בן עם אותו ערך מפתח.

ה. **LCHILD**: פקודה זו מיועדת להגדרת אינדקס או קשר לוגי של הסגמנט.

ו. **END, FINISH, DBDGEN**: פקודות המסיימות את הגדרת בסיס הנתונים.

תרשים 35—11 מציג בסיס נתונים לדוגמא ובהמשך ניתנת ההגדרה שלו.

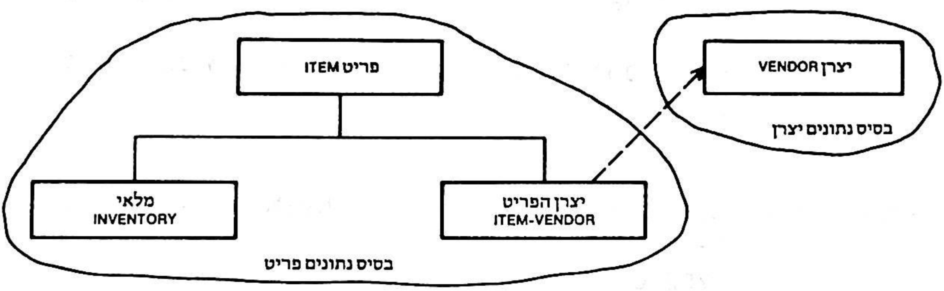


תרשים 35 - 11: בסיס נתונים

DBD	NAME=INVENTORY—DB,	שם בסיס הנתונים
	ACCESS=HDAM	שיטת ארגון HDAM
DATASET	DD1= INV,	שם הקובץ
	DEVICE=3340,	סוג הדיסק
	BLOCK=2048	גודל הגוש בבתים
SEGM	NAME=ITEM,	שם סגמנט הפריט
	PARENT=0,	הגדרת סגמנט שורש (o)
	BYTES=9	אורך הסגמנט
FIELD	NAME=(ITEM-NO,SEQ,U),	שדה המפתח
	BYTES=9,	אורך השדה
	TYPE=C	שדה אלפא־נומרי (c)
SEGM	NAME=INVENTORY,	שם סגמנט המלאי
	PARENT=ITEM,	שם סגמנט אב
	DBLE,	מצביע לבן הראשון והאחרון
	BYTES=16	אורך הסגמנט
FIELD	NAME=QUANTITY,	שדה בסגמנט INVENTORY
	BYTES=10,	
	TYPE=C	
FIELD	NAME=DATE,	שדה בסגמנט INVENTORY
	BYTES=6,	
	TYPE=C	
SEGM	NAME=VENDOR,	שם סגמנט יצרן
	PARENT=ITEM,	שם סגמנט אב
	BYTES=51,	
	POINTER=TWINBD	מצביעים PTF, PTB
FIELD	NAME=(VEND-NO, SEQ, M),	שדה מפתח בסגמנט VENDOR
	BYTES=5,	
	TYPE=C	
FIELD	NAME=VEND-NAME,	שדה בסגמנט VENDOR
	BYTES=40,	
	TYPE=C	

FIELD	NAME=DATE, BYTES=6, TYPE=C	שדה בסגמנט VENDOR
DBDGEN FINISH END		הוראות סיום ההגדרה

הדוגמא בתרשים 36—11 מציגה הגדרת קשר לוגי בין שני בסיסי נתונים. קשר לוגי זה מייצג למעשה קשר רב—רב־ערכי בין הפריט לבין היצרן. שני הסגמנטים, סגמנט ITEM וסגמנט VENDOR הם סגמנטי אב לסגמנט ITEM-VENDOR. שיטה זו מאפשרת להחזיק את נתוני היצרן בבסיס נתונים יצרן בלבד ובבסיס נתונים פריט להשתמש במצביע מהבן הלוגי ITEM-VENDOR אל האב הלוגי VENDOR. בזמן הגדרת בסיס נתונים יצרן מציינים באמצעות פקודת LCHILD את העובדה שיש לו בנים לוגיים.



תרשים 36 - 11: קשר לוגי בין שני בסיסי נתונים

הסכימה (DBD) עבור המבנה שבתרשים 36—11:

DBD	NAME=INVENTORY-DB, ACCESS=HDAM	בסיס נתונים
DATASET	(Parameters)	קובץ פיסי
SEGM	NAME=ITEM, PARENT=0, BYTES=9	סגמנט שורש
FIELD	NAME=(ITEM-NO, SEQ,U), BYTES=9, TYPE=C	שדה מפתח
SEGM	NAME=INVENTORY, PARENT=ITEM, BYTES=16	סגמנט
FIELD	NAME=QUANTITY, BYTES=10, TYPE=C	שדה
FIELD	NAME=DATE, BYTES=6, TYPE=C	שדה
SEGM	NAME=ITEM-VENDOR, POINTER=LPARENT	סגמנט עם קשר
	PARENT=((ITEM), (VENDOR, VEND-DB)), BYTES=51	אב לוגי

FIELD	NAME=(VEND-NO, SEQ, M), BYTES=5, TYPE=C	שדה
FIELD	NAME=DATE, BYTES=6, TYPE=C	שדה
DBDGEN		סיום ההגדרה
FINISH		
END		

DBD	NAME=VEND-DB, ACCESS=HDAM	בסיס נתונים
DATASET	(Parameters)	קובץ פיסי
SEGM	NAME=VENDOR, BYTES=85	סגמנט
LCHILD	NAME=(ITEM-VENDOR, INVENTORY-DB)	בן לוגי
FIELD	NAME=(VEND-NUMBER, SEQ, U), BYTES=5	שדה מפתח
FIELD	NAME=VEND-NAME, BYTES=40	שדה
FIELD	NAME=VEND-ADDRESS, BYTES=40	שדה
DBDGEN		סיום ההגדרה
FINISH		
END		

5.2 הגדרת תת־סכימה

לכל תוכנית המשתמשת בבסיס הנתונים מוגדרת תת־סכימה (PSB—Program Specification Block). תת־הסכימה היא אוסף של **שטחי תקשורת** (PCB — Program Communication Block) בין מערכת IMS לבין תוכנית היישום המשתמשת בבסיס הנתונים. עבור כל בסיס נתונים, פיסי או לוגי, יש להגדיר שטח תקשורת (PCB) אחד אשר מכיל תיאור של חלק מהסגמנטים שהוגדרו בבסיס הנתונים ואשר תוכנית היישום זקוקה להם. שטח התקשורת מגדיר למעשה תת־עץ במבנה ההירככי ולכן הוא חייב להתחיל בסגמנט השורש. ניתן להגדיר מספר בלתי מוגבל של שטחי תקשורת לאותו בסיס נתונים ומספר בלתי מוגבל של תוכניות יישום רשאיות להשתמש באותו שטח באמצעות תת־סכימות (PSB) זהות או שונות.

תיאור קצר של השדות המרכיבים את **שטח התקשורת**, PCB:

א. **DBD-NAME**: מכיל את שם בסיס הנתונים כפי שהוגדר בסכימה (DBD).
ב. **SEG-LEVEL**: מכיל את הרמה ההירככית של הסגמנט שטופל בפקודת DML אחרונה. סגמנט השורש נמצא ברמה 1, הבנים שלו ברמה 2 וכך הלאה.
ג. **STATUS-CODE**: מכיל **קוד החזר** (Return Code) המציין אם הושלם הביצוע של פקודת DML האחרונה. אם הביצוע לא הושלם כהלכה, יכיל השדה את קוד השגיאה.
ד. **PROC-OPTIONS**: מכיל את אופציות העיבוד המותרות עבור הסגמנט האחרון שטופל.
ה. **SEG-NAME**: מכיל את שם הסגמנט האחרון שטופל.
ו. **FB-KEY-LENGTH**: מכיל את האורך הנוכחי של המפתח ההירככי המשורשר עבור הסגמנט האחרון שטופל.
ז. **FB-KEY-AREA**: מכיל את המפתח ההירככי המשורש של הסגמנט האחרון שטופל.

לדוגמא, בגישה לסגמנט היצרן בבסיס נתונים פריט המופיע בתרשים 36—11 יכיל המפתח המשורשר את מספר הפריט ואת מספר היצרן.

השפה להגדרת תת־הסכימה מורכבת ממספר פקודות מקרו כאשר לכל אחת מהן יש מספר פרמטרים. להלן תיאור קצר של הפקודות המשמשות להגדרת **תת־הסכימה:**

א. **PCB**: פקודה זו קובעת באמצעות פרמטר DBDNAME את שם בסיס הנתונים שהתוכנית תשתמש בו. פרמטרים נוספים כוללים את אופציות העיבוד, אורך המפתח ועוד.

הפרמטר PROCOPT מאפשר להגדיר את אופציות העיבוד המותרות לתוכנית היישום. האופציות הנפוצות הן G (Get) עבור קריאה בלבד, D (Delete) עבור ביטול, I (Insert) עבור הוספה, R (Replace) עבור עדכון. האופציה K (Key) מציינת שתוכנית היישום אינה יכולה לטפל בסגמנט, אבל הוא מהווה חלק מתת־הסכימה והיא יכולה להשתמש בסגמנטים הבנים שלו.

ב. **SENSEG**: כדי שתוכנית יישום תוכל לפנות לסגמנט יש להגדירו בפקודה זו. את הסגמנטים יש לרשום באותו סדר שבו נרשמו הסגמנטים במבנה ההירכי ב־DBD המתאים.

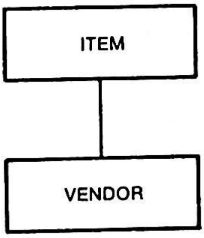
ג. **SENFLD**: מאפשרת את הגדרת השדה בתוך הסגמנט כדי שהתוכנית תוכל לקרוא אותו או לעדכנו. פקודה זו מאפשרת להגביל את התוכנית לטיפול רק בחלק מהשדות של הסגמנט. אם פקודה זו אינה מופיעה בהגדרה, DL/1 מניח שהתוכנית יכולה לטפל בכל השדות הסגמנט.

ד. **VIRFLD**: הפקודה מאפשרת הגדרת שדה בפועל, שדה שאינו מופיע בסגמנט הפיסי. בין הפרמטרים של הפקודה אפשר לכלול שם שגרה אשר תקרא אוטומטית מספריית התוכניות כאשר תוכנית היישום תפנה לשדה זה.

ה. **PSBGEN**: מציין את סוף הגדרת תת־הסכימה (PSB) וקובע את שפת התכנות של תוכנית היישום ואת שם תת־הסכימה.

ו. **END**: מציין את גמר ההגדרה.

לכל תוכנית המשתמשת בבסיס הנתונים יש לכתוב תת־סכימה, כדוגמא, נכתוב תת־סכימה (PSB) עבור תוכנית שצריכה להפיק דוח אשר מציג את רשימת היצרנים לכל פריט. המבנה הלוגי של הנתונים ניתן בתרשים 37—11, והגדרת הסכימה ניתנת בהמשך.



תרשים 37 - 11: מבנה לוגי של נתונים לתוכנית יישום לדוגמא

PCB	TYPE=DB,	
	DBNAME=INVENTORY-DB,	שם בסיס הנתונים
	PROCOPT=G	אופצית העיבוד המותרת: קריאה בלבד
	KEYLEN=M	אורך המפתח המשורשר של שני הסגמנטים
SENSEG	NAME=ITEM,	סגמנט שורש
	PARENT=0	
SENSEG	NAME=VENDOR,	סגמנט
	PARENT=ITEM	שם סגמנט אב
SENFLD	NAME=VEND-NO	שדה
SENFLD	NAME=VEND-NAME	שדה
PSBGEN	LANG=COBOL	שפת תכנות קובול
	PSBNAME=REP-104	שם תת־הסכימה
END		

6. שפה לטיפול בנתונים

מערכת DL/1 כוללת שפה לטיפול בנתונים (DML — Data Management Language) אשר משולבת בתוך שפה מארחת כגון COBOL, PL/1 או ASSEMBLER. הטיפול יכול להיות בסגמנט אחד או יותר בו־זמנית, ולכן צריך לנהל מראי מקום, אשר מציינים את מיקום תוכנית היישום בתוך בסיס הנתונים.

6.1 מבנה פקודת DML

הפנייה ל־ DL/1 בקובול למשל מתבצעת באמצעות הוראת CALL בעלת המבנה הבא:

CALL ‘CBLTDLI’ USING function, PCB-name, I/O Area, SSA_1, ...SSAn.

— **function**: הגדרת הפעולה אשר תוכנית היישום מבקשת ש־ DL/1 תבצע:

GU — (Get Unique): שליפת סגמנט בהתאם לתנאי הנקבע ב־SSA. השליפה אינה תלוייה במיקום התוכנית בבסיס הנתונים. פקודה זו מאפשרת הצבה ראשונית לצורך התחלת העיבוד, או מעבר לשלב עיבוד אחר.

GN — (Get Next): שליפת הסגמנט הבא לפי המיקום הנוכחי של התוכנית. הסגמנטים נשלפים לפי הסדר ההירכי, מעלה־מטה־שמאל־ימין. אם קיים SSA ישלפו רק הסגמנטים שמתאימים לתנאי השליפה.

GNP — (Get Next Within Parent): שליפת כל הסגמנטים השייכים לרשומת אב אחת. סגמנט האב נקבע על ידי פקודת GU או GN אחרונה. אם קיים SSA ישלפו רק הסגמנטים שמתאימים לתנאי השליפה.

GH — (Get Hold): פקודה זו מאותתת ל־DL/1 שהתוכנית מתכוונת לעדכן את הסגמנט שנשלף ועליה למנוע טיפול בו על ידי תוכניות אחרות. הוראת HOLD (H) אפשר לשלב בכל אחת מהפקודות הקודמות: GHU, GHN, GHNP.

ISRT — (Insert): זוהי פקודה להוספת סגמנט חדש לבסיס הנתונים. יש לקבוע היכן להוסיף את הסגמנט באמצעות SSA מתאים.

DLET — (Delete): פקודה לביטול הסגמנט וביטול כל הסגמנטים התלויים בו. פקודה זו יכולה לגרום לפגיעה בבסיס הנתונים ועל כן יש להגביל את השימוש בה.

REPL — (Replace): פקודה לעדכון תוכן סגמנט אשר קיים בבסיס הנתונים. אין אפשרות לשנות את מפתח הסגמנט, אלא על ידי ביטול הסגמנט והוספת סגמנט חדש במקומו.

— **PCB-name**: קובע את שם תת־הסכימה שהתוכנית משתמשת בה. תת־הסכימה מכילה שטח תקשורת (PCB) בין התוכנית לבין DL/1. שטח התקשורת מכיל נתונים שונים, כגון:

- שם בסיס הנתונים כפי שנקבע בסכימה (DBD),
- סטטוס של הפקודה שבוצעה. על תוכנית היישום לבדוק שדה זה לאחר כל פעולה,
- סוגי הפעולות המותרים לתוכנית (קריאה, כתיבה, עדכון),
- הרמה ההירכית של הסגמנט שטופל בפעולה אחרונה,
- אורך המפתח המשורשר שהוחזר,
- המפתח המשורשר של כל הסגמנטים לאורך המסלול ההירכי.

— **I/O Area**: שטח בתוך תוכנית היישום שאליו או ממנו יקראו או יכתבו הסגמנטים על ידי מערכת DL/1.

— **SSA** (Segment Search Argument): זהו **תנאי לשליפת סגמנט**. אפשר לציין ארגומנט שליפה לכל סוג סגמנט במבנה ההירכי, אך סדר כתיבתם צריך להיות לפי סדר הסריקה של המבנה. את ארגומנט השליפה כותבים בצורה הבאה:

Segment-name, Command-code, Qualification

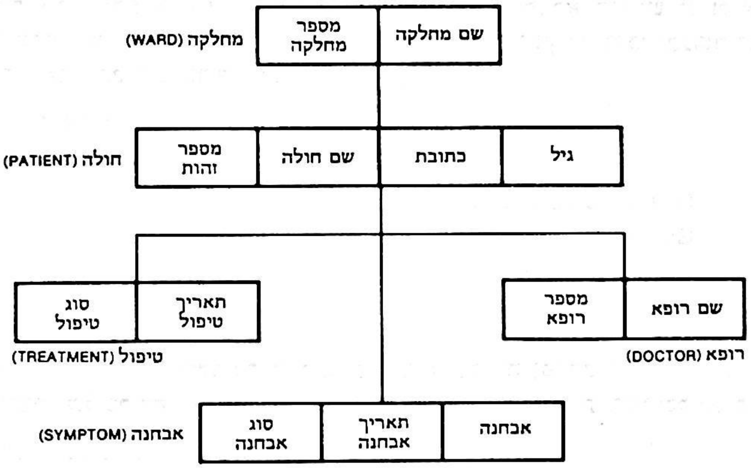
ניתן לזהות את הסגמנט על ידי ציון שמו או על ידי ציון שמו וערך של שדה בתוך הסגמנט. **תו ההוראה** (Command-Code) הינו אופציונלי ומאפשר שינוי מסויים בפקודת DL/1 שבהוראה כפי שנראה בסעיף הבא. את **התנאי** (Qualification) רושמים בצורה הבאה:

Field-name, Condition, Value

בדוגמאות שבהמשך נרשום לשם הפשטות את תנאי השליפה (SSA) בצורה סימבולית ולא בצורה פורמלית, כפי שהוא צריך להכתב.

6.2 דוגמאות לפקודות DML

נציג מספר דוגמאות למבנה פקודות DML במערכת DL/1 עבור המבנה ההיררכי שבתרשים 38 -- 11.



תרשים 38 - 11: בסיס נתונים היררכי

א. גישה ישירה (Direct Retrieval):

1. שלוף את נתוני מחלקה 702:

```
GET UNIQUE WARD (WARD-NO=702).
```

2. שלוף את נתוני חולה 8425 המאושפז במחלקה 604:

```
GET UNIQUE WARD (WARD-NO=604).
  PATIENT (PATIENT-NO=8425)
```

3. שלוף את נתוני האבחנה של החולה בדוגמא 2 שנעשתה בתאריך שלאחר 01/04/82:

```
GET UNIQUE  WARD (WARD-NO=604)
            PATIENT (PATIENT-NO=8425)
            SYMPTOM (SYM-DATE>820401).
```

ב. שליפה סדרתית (Sequential Retrieval):

1. שלוף את נתוני כל הרופאים שטיפלו בחולה שמספרו 8425:

```
           GET UNIQUE  WARD
                       PATIENT (PATIENT-NO=8425)
                       DOCTOR.
LABEL-1.   GET NEXT DOCTOR.
```

בדוק אם מפתח הירָרכי משורשר מכיל מספר חולה 8425, אם כן בצע עיבוד וחזור לכתובת LABEL-1. אחרת המשך.

בדוגמא זו לא קבענו את מספר המחלקה, שהוא המפתח של סגמנט השורש. לכן תבצע מערכת DL/1 סריקה סדרתית בבסיס הנתונים עד לאיתור סגמנט חולה ראשון עם המספר המבוקש. כל פקודת GET NEXT תשלוף סגמנט רופא נוסף. לאחריה יש לבדוק אם התחלף סגמנט החולה מכיון שהפקודה עלולה לחצות את גבולות העץ ההירַרכי ולעבור לחולה אחר (הוראה GNP תחסום מעבר לרשומת אב אחרת).

2. שלוף את כל הסגמנטים המופיעים בבסיס הנתונים:

```
           GET UNIQUE WARD.
LABEL-1.   GET NEXT.
           GO TO LABEL-1.
```

בדוגמא זו מתבצעת סריקה סדרתית של כל בסיס הנתונים לפי הסדר ההירַרכי (מעלה־מטה־שמאל־ימין) של כל עץ. משתמשים בצורת שליפה זו לשם העתקת גיבוי של בסיס הנתונים.

3. שלוף את נתוני האבחנות של חולים שגילם נע בין 25 ל־30:

```
          GET  UNIQUE WARD
               PATIENT (AGE ≥ 25 AND AGE ≤ 30)
               SYMPTOM.
          MOVE PATIENT-NO TO TEMP-PATIENT-NO.
LABEL-2.  GET NEXT SYMPTOM.
         IF PATIENT-NO (PSB המפתח משורשר ב־) = TEMP-PATIENT-NO
              GO TO LABEL-2.
```

ג. שליפה סדרתית בתוך רשומת אב (Gequential Retrieval within Parent)

השימוש בפקודה GNP דומה לשימוש בפקודה GN. ההבדל הוא שפקודת GNP מחזירה לשטח התקשורת קוד־החזר מיוחד לאחר שהיא מסיימת לסרוק את כל הסגמנטים מאותו סוג בתוך רשומת האב שנקבעה על ידי פקודת GU או GN קודמת. בפקודת GN אשר יכולה לחצות גבולות של רשומת אב ללא התרעה הבדיקה חייבת להעשות על ידי תוכנית היישום.

ד. עדכון סגמנט

כדי לעדכן סגמנט יש לשלוף אותו תחילה באמצעות אחת מפקודות השליפה ולציין את אופציית HOLD.

— עדכן את שם החולה:

```
GET HOLD UNIQUE WARD
            PATIENT (PATIENT-NO=8425).
MOVE NEW-NAME TO PATIENT-NAME.
REPLACE.
```

ה. הוספת סגמנט חדש

פקודת ההוספה חייבת להגדיר את המסלול ההירככי עד לסגמנט האב של הסגמנט החדש. מערכת DL/1 תוסיף את הסגמנט החדש מתחת לסגמנט האב שנבחר ובסדר המתאים בהתאם לתוכן שדה המפתח.

— הוסף רשומת אבחנה חדשה לחולה שמספרו 8425:

```
INSERT  WARD
        PATIENT (PATIENT-NO=8425)
        SYMPTOM.
```

ו. ביטול סגמנט קיים

בדומה לעדכון סגמנט, גם בביטול יש תחילה לשלוף את הסגמנט באמצעות אחת מפקודות השליפה עם האופציה HOLD.

— בטל רשומת חולה שמספרו 8425 יחד עם כל הסגמנטים התלויים:

```
GET HOLD UNIQUE WARD
      PATIENT (PATIENT-NO=8425).
DELETE.
```

ז. השימוש בתו הוראה (Command Code)

תנאי שליפת הסגמנט, SSA, יכול להכיל תו מיוחד המאפשר הרחבה מסויימת של פקודות DML. נסקור בקצרה חלק מתווי ההוראה האפשריים.

1. שליפה הירככית (Command Code = D)

אופציה זו מאפשרת שליפת מספר סגמנטים באמצעות פקודות DL/1 אחת. בפקודה הבאה למשל נרשמה האופציה D ולכן ישלפו גם סגמנט המחלקה וגם סגמנט החולה בפקודה אחת. הפקודה ללא אופציה D היתה שולפת רק סגמנט אחד כפי שנדרש בתנאי לשליפת סגמנט החולה.

```
GET UNIQUE WARD*D
PATIENT (PATIENT-NO=8425).
```

אופציה D מיועדת לשליפת מספר סגמנטים במסלול ההירככי בהתאם לבחירת המשתמש וסגמנטים אחרים לא ישלפו. אופציה D יכולה להופיע ברמה הירככית אחת או יותר בהתאם לצורך, ואין צורך לציין אותה ברמה ההירככית האחרונה המופיעה ברשימת התנאים לשליפה כי זו תשלף תמיד.

ניתן לשלב אופציה זו עם פקודות INSERT, DELETE, REPLACE.

2. סריקה מהתחלה (Command Code = F)

במקרים מסויימים עורכים סריקה לפנים במבנה ההירכי של הרשומה ולאחר שליפת סגמנט מסויים ובדיקתו יש צורך לשלוף סגמנט הנמצא לפניו בסדר ההירכי. שימוש בפקודה GET NEXT מאפשר סריקה רק בכיוון לפנים ועל כן נזקקים להוראת חזרה כביכול. אופציה F מורה שהחיפוש אחר הסגמנט הזה יתחיל מהסגמנט האחרון שנקרא באמצעות פקודה GET UNIQUE או GET NEXT.

יש לבדוק למשל את סוגי האבחנה של החולה. אם מזהים סוג אבחנה מסויים צריך לשלוף את כל הסגמנטים של הטיפולים של אותו חולה. מכיון שסגמנט הטיפול נמצא בסדר ההירכי לפני סגמנט האבחנה, לא ניתן להשתמש בפקודת שליפה רגילה ויש להוסיף לה את ההוראה לסרוק מההתחלה:

```
           GET UNIQUE WARD (WARD-NO=508).
LABEL-1.   GET NEXT WITHIN PARENT PATIENT SYMPTOM.
           IF NOT SYMPTOM-TYPE-ABC
                GO TO LABEL-1.
           GET NEXT WITHIN PARENT TREATMENT ★F.
```

3. גישה לסגמנט אחרון (Command Code=L)

כאשר יש צורך לגשת לסגמנט אחרון מסוג כלשהו אפשר לעשות זאת בפקודה אחת אם מציינים את קוד ההוראה L. אם הגדרת בסיס הנתונים (DBD) הכילה מצביע לסגמנט אחרון מבין התאומים בסוג הסגמנט, תתבצע גישה מיידית. אחרת — תבצע המערכת סריקה לפנים עד לקבלת הסגמנט האחרון הרצוי.

הפקודה לשליפת סגמנט הטיפול האחרון של החולה שבו אנו עוסקים:

```
GET UNIQUE WARD
           PATIENT
           TREATMENT★L.
```

7. סיכום

מערכת ניהול בסיסי נתונים IMS מטפלת בניהול נתונים בעלי מבנים מורכבים ביותר. המערכת מונעת כפילויות ומאפשרת גמישות רבה בהגדרת הקשרים הלוגיים שבין בסיסי הנתונים הפיסיים. קיימת אפשרות להגדיר צירופים שונים של מצביעים בין הסגמנטים ולמעשה ניתן להגדיר מבנים דומים לרשת למרות שצורת הטיפול היא תמיד הירכית.

השימוש ב־IMS מחייב להגדיר במפורש את השטחים לשימוש על־ידי DL/1. הוראות קלט/פלט במערכת מחייבים ידע של המבנים הלוגיים.

מערכת IMS הינה אחת ממערכות DBMS הגדולות והמקיפות ביותר הקיימות היום. מכיוון שהיא פותחה בסוף שנות ה־60 היא נמצאת בשימוש נרחב מאד במתקנים של מחשבי יבמ גדולים. הארכיטקטורה של המערכת והמונחים המקובלים בה הם יחודיים ואינם תואמים את המלצות CODASYL DBTG, אשר הוגשו לאחר ש־IMS כבר פעלה במתקנים רבים.

כפי שנאמר בתחילת פרק זה, קיימת מערכת ניהול בסיסי נתונים דומה DOS/VS-DL/1 אשר מופעלת במחשבי יבמ בינוניים וקטנים בסביבת מערכת ההפעלה DOS. יש הבדלים קטנים בינה לבין רכיב DL/1 של IMS (IMS/DB).

שירותי מילון נתונים אפשריים לשתי מערכות אלו באמצעות מוצר תוכנה נפרד ולא נסקור אותו כאן.

פרק יב

IDMS — Integrated Database Management System

1. מבוא
2. מושגי יסוד
3. מבנים פיסיים
4. מבנים לוגיים של נתונים
5. שפה להגדרת נתונים
6. שפה לטיפול בנתונים
7. סיכום

1. מבוא

מערכת ניהול בסיסי נתונים IDMS פותחה על ידי חברת Cullinet Corporation בשנת 1975 בהתאם להמלצות CODASYL DBTG. המערכת מיועדת לפעול במחשבי יבמ הבינוניים והגדולים. המערכת מכילה שפה להגדרת נתונים אשר נפרדת מתוכניות היישום, שפה להגדרת המבנה הפיסי ושפה לטיפול בנתונים. המערכת מכילה מילון נתונים המהווה חלק בלתי נפרד ממנה.

2. מושגי יסוד

בתחילת הדיון נסקור את מושגי היסוד המקובלים במערכות IDMS:

— **שדה** (Data Item): יחידת הנתונים הבסיסית והקטנה ביותר במערכת. לכל שדה יש שם סימבולי אשר מאפשר לתוכנית היישום לפנות אליו.

— **רשומה** (Record): אוסף של מספר שדות בעלי משמעות לוגית מבחינת המשתמש. הרשומה הינה יחידת התקשורת הבסיסית בין בסיס הנתונים לבין תוכנית היישום.

— **סוג רשומה** (Record Type): אוסף בלתי מוגבל של מופעים של רשומות מאותו סוג. ניתן להגדיר מספר בלתי מוגבל של סוגי רשומות שונים אשר לכל אחד מהם נקבע שם ייחודי.

— **חבורה** (Set): החבורה היא אוסף של סוגי רשומות המורכבות מסוג רשומה אחד שהוא **רשומת אב** (Owner Record Type) ואחד או יותר סוגי רשומות שהם **רשומות בן** (Member Record Type). לכל חבורה שם יחודי ויכולים להיות לה מופעים אחדים. כל מופע של חבורה מורכב ממופע אחד של רשומת אב ואפס, אחד או יותר מופעים של רשומות בן.

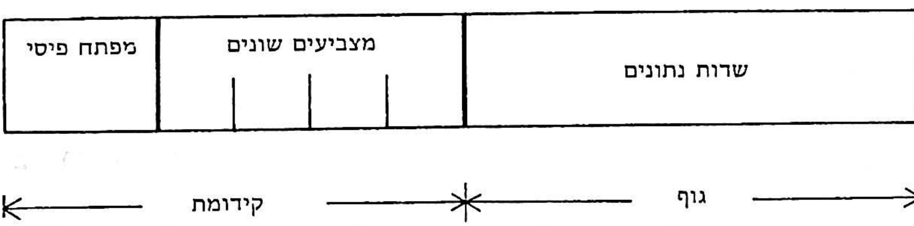
— **קובץ** (File): הקובץ מוגדר כמבנה פיסי המוכר למערכת ההפעלה. הוא יכול להכיל מופע של סוג רשומה אחד או יותר ושל הקשרים שבין סוגי הרשומות.

— **בסיס הנתונים** (Data Base): אוסף כל הקבצים המוגדרים בסכימה אחת.

3. מבנים פיסיים

כל הקבצים של מערכת IDMS הם קבצים בגישה ישירה, מיוצגים בשיטת BDAM של יבמ. במבנה הפיסי מרכיבים אחדים:

— **רשומה**. לכל רשומה שני חלקים: **קידומת** (Prefix) המכילה את המפתח הפיסי ואת המצביעים השונים הקושרים אותה לחבורות שבהן היא מופיעה, **גוף** (Body) המכיל את כל שדות הנתונים של הרשומה. לכל רשומה בבסיס הנתונים יש **מפתח פיסי** (Data Base Key) המזהה אותה בצורה חד־משמעית ואשר אינו ניתן לשינוי. המצביעים בקידומת הרשומה הם המפתחות הפיסיים של הרשומות שרשומה זו קשורה אליהם.



תרשים 1 – 12: מבנה רשומה ב־IDMS

— **דפים** (Pages). בסיס הנתונים מחולק לדפים בעלי מספרי זיהוי. כל דף יכול להכיל מספר רשומות בהתאם לגדלו, ובמקרה של רשומות באורך משתנה הן יכולות להתפרס על פני מספר דפים. הדף מחולק **לשורות** אשר כל אחת מהן מכילה מופע אחד של רשומה (באורך קבוע). בשיטה זו מורכב המפתח הפיסי של הרשומה ממספר הדף ומספר השורה שבו היא מאוחסנת. המבנה הלוגי של בסיס הנתונים מיושם באמצעות המצביעים הקושרים את הרשומה אל רשומות אחרות. המיקום הפיסי של הרשומה בלתי תלוי במבנה הלוגי המוגדר על־ידי החבורות.

— **שטחים** (Areas). השטח הינו חלוקה לוגית של בסיס הנתונים. כל שטח מורכב מדפים אשר מכילים את הרשומות. בסיס הנתונים מורכב מאחד או יותר שטחים בהתחשב באילוצים הבאים:

א. כל דף בבסיס הנתונים יכול להשתייך לשטח אחד בלבד ואין אפשרות לחפיפה בין שטחים,

ב. כל הדפים בשטח אחד הם בעלי גודל פיסי זהה,

ג. כל המופעים של רשומות מאותו סוג חייבים להופיע בשטח אחד,

ד. על המשתמש לקבוע טווח מספרים לזיהוי הדפים בשטח. כל הדפים השייכים לשטח אחד חייבים להיות בטווח רצוף וסדרתי של מספרים. ניתן ליצור מרווחים בספרור הדפים בין השטחים השונים.

— **קבצים** (Files). בסיס הנתונים מאוחסן באחד או יותר קבצים לגישה ישירה. כל קובץ מורכב מאוסף של **גושים פיסיים** (Blocks) כאשר בכל אחד מהם דף לוגי אחד. IDMS מאפשר להקצות את השטחים לקבצים לפי אחת מהאפשרויות הבאות:

א. כל בסיס הנתונים יאוחסן בקובץ אחד,

ב. כל שטח יופיע בקובץ נפרד,

ג. מספר שטחים שונים יאוחסנו בקובץ אחד,
ד. שטח אחד יאוחסן במספר קבצים שונים.

3.1 שיטות איתור רשומות (Location Mode)

קיימות שלש שיטות לאחסון ולאיתור רשומות בבסיס הנתונים:

— **CALC**: בשיטה זו המשתמש קובע שדה אחד מתוך הרשומה כשדה מפתח — זהו המפתח הלוגי של הרשומה. מערכת IDMS מפעילה תהליך חישובי לתרגום המפתח הלוגי לכתובת פיסית. התהליך החישובי "דואג" לפזר את הרשומות בצורה אחידה עד כמה שניתן בשטח שבו מאוחסנות הרשומות מאותו סוג.

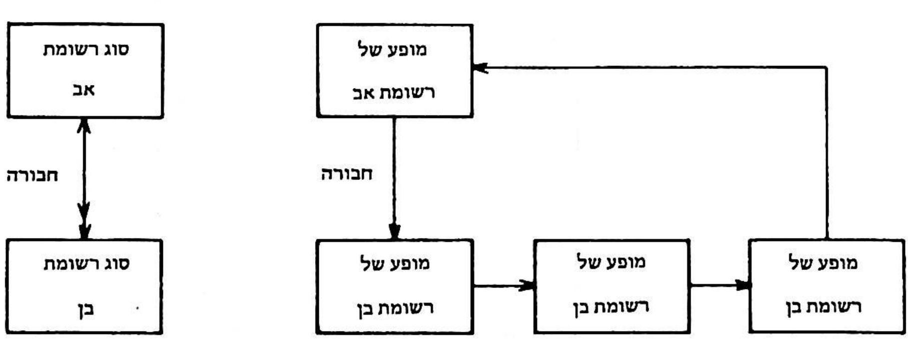
שיטת CALC מאפשרת למשתמש גישה ישירה לרשומה על פי תוכן המפתח הלוגי שלה. מפתחות לוגיים אינם בהכרח יחודיים מבחינת הכתובות הפיסיות ולכן אפשרי מצב של כתובות כפולות (Duplicates) אשר המשתמש צריך לציין כיצד לטפל בהן.

— **VIA**: ניתן להגדיר שיטת איתור VIA אם הרשומה משתתפת כרשומת בן בחבורה כלשהי. רשומות הבן השונות מאוחסנות באותו דף שבו מאוחסנת רשומת האב בחבורה. אם אין מקום אחסון פנוי תנסה המערכת לאחסן את הרשומה בדף הבא וכך הלאה, עד למציאת מקום אחסון פנוי וכתוצאה מכך הגישה היא סודרת ואינה מתאימה לגישה ישירה. שיטה זו יכולה להעלות את יעילות העיבוד של בסיס הנתונים מאחר וניתן לשלוף את המופע של החבורה בהוראת קלט אחת. מקובל לקבוע שיטת VIA לרשומת בן בחבורה שבה תדירות הגישה מרשומות האב לרשומות הבן היא הגבוהה ביותר.

— **DIRECT**: הרשומה מאוחסנת בדף ובשורה אשר נקבעים על ידי המשתמש. לשליפת הרשומה על המשתמש לדעת את המפתח הפיסי שלה. שיטת איתור זו מחייבת ידיעת המיקום הפיסי של הרשומה ולכן משתמשים בה למטרות מיוחדות בלבד.

3.2 חבורות (Sets)

הרשומות ב־IDMS קשורות יחדיו למבנה לוגי הנקרא חבורה. לכל חבורה יש להגדיר סוג רשומה אחד כרשומת אב ואחד או יותר סוגי רשומות המוגדרות כרשומות בן. מופע של חבורה מורכב ממופע אחד של רשומת אב ומספר כלשהו של מופעים של רשומות בן.



תרשים 2 – 12: מבנה חבורות

אפשר להגדיר את סוג הקשר בין הרשומות בחבורה. בכל הרשומות מופיע מצביע NEXT אל הרשומה הבאה בחבורה והוא מאפשר עיבוד סדרתי שלה. כל רשומה מכילה מצביע NEXT אחד עבור כל חבורה שבה הרשומה מוגדרת כרשומת אב או כרשומת בן. אם יש צורך בסריקת החבורה בכיוון הפוך מגדירים מצביע PRIOR אל הרשומה הקודמת בחבורה. ניתן לקשור את רשומות הבן באופן ישיר לרשומת האב אם מגדירים מצביע OWNER. המצביעים PRIOR ו־OWNER הם אופציונליים.

נזכור שכל המצביעים מכילים למעשה את המפתחות הפיסיים של הרשומות בבסיס הנתונים. הם מאוחסנים בקידומת של כל רשומה ומאפשרים לה להשתתף ברשומת אב או כרשומת בן בחבורות שונות. הגדרת הרשומה כוללת את המיקום היחסי של כל אחד מהמצביעים השונים בקידומת הרשומה.

4. מבנים לוגיים של נתונים

המבנים הלוגיים ב־IDMS הם כל המבנים שניתן לבנות באמצעות החבורות. מבנים אלה הוסברו בפרק המתאר את מודל הרשת ולכן לא נחזור עליהם כאן.

5. שפה להגדרת נתונים

סכימות ותת־סכימות במערכת IDMS מוגדרות באמצעות שפה להגדרת נתונים (DDL) שכתיבתה דומה לכתיבה בקובול. הסכימה מגדירה את בסיס הנתונים, ותת־הסכימות מאפשרות לתוכנית היישום השונות את הגישה לבסיס הנתונים. הסכימה ותת־הסכימות מאוחסנות במילון הנתונים של IDMS.

5.1 שפה להגדרת הסכימה

הסכימה מוגדרת באמצעות חמש קבוצות של משפטים:

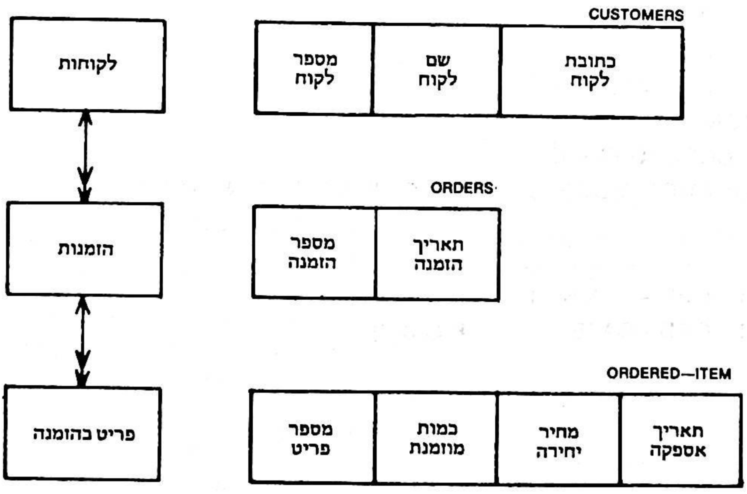
— **Schema Description**: במשפט זה קובע מנהל בסיס הנתונים את שם הסכימה. הוא יכול להוסיף נתונים כגון תאריך היצירה, שם המתכנת והערות נוספות.

— **File Description**: במשפטים אלה מגדירים את הקישור בין הקבצים של בסיס הנתונים לבין הקבצים הפיסיים וקובעים את הסוג של כל קובץ. על המשתמש להגדיר לפחות שני קבצים: קובץ אחד לשם **רישום האירועים** (Journal) וקובץ אחד שיכיל את **בסיס הנתונים**. אפשר לאחסן את בסיס הנתונים במספר קבצים פיסיים שונים.

— **Area Description**: במשפטים אלה קובעים את שמות השטחים ואת טווח מספרי הדפים שיאוחסנו בכל שטח. בנוסף נקבע גם המיפוי בין הדפים הלוגיים לבין הגושים הפיסיים בקבצים. מספר הדפים המוקצה לשטח קובע את גודלו.

— **Record Description**: במשפטים אלה מציינים את סוגי הרשומות בסכימה, את המופע שלהם בשטחים השונים ועוד. תיאור סוג הרשומה כולל את שם הרשומה, מזהה יחודי של סוג הרשומה, צורת האיתור ושדות הנתונים שלה.

— **Set Description**: במשפטים אלה מגדירים את הקשרים בין סוגי הרשומות שהוגדרו בסכימה. לכל חבורה נקבע שם, צורת קישור הרשומות וציון רשומות האב ורשומות הבן, סדר הרשומות בחבורה וסוג החברות.



תרשים 3 – 12: סכימה

את הסכימה שבתרשים 12-3 יש להגדיר בצורה הבאה:

```
SCHEMA DESCRIPTION.
   SCHEMA NAME IS CUST—ORDERS.                                        שם בסיס הנתונים
FILE DESCRIPTION.                                                     תיאור הקבצים
   FILE NAME IS DB-FILE            ASSIGN TO SYS010
                                   DEVICE TYPE IS 3330.
   FILE NAME IS JOURNAL            ASSIGN TO SYS009.
AREA DESCRIPTION.                                                     מספרי הדפים
   AREA NAME IS CUST—AREA          RANGE 2001 THRU 3000               בשטח
                                   WITHIN FILE DB-FILE
                                   FROM 1 TO 1000.                    מספרי דפים יחסיים
   AREA NAME IS ORD—AREA           RANGE 5001 THRU 7000
                                   WITHIN FILE DB–FILE
                                   FROM 1001 TO 3000.
RECORD DESCRIPTION.
   RECORD NAME IS CUSTOMER                                            רשומת לקוח
   RECORD ID IS 300                                                   זיהוי הרשומה
   LOCATION MODE IS CALC USING CUST—ID                                מפתח הרשומה
                             DUPLICATES NOT ALLOWED                   וצורת האיתור
                             WITHIN CUST—AREA
                             CALL CHECK—CUST BEFORE STORE.
                                                                      שגרה לבדיקה לפני אחסון
      03  CUST—ID               PIC X(5).
      03  CUST—NAME             PIC X(30)
      03  CUST—ADDRESS.
          05  CITY              PIC X(20).
          05  STREET            PIC X(10).
          05  NUMBER            PIC X(5).
   RECORD NAME IS ORDER
      RECORD ID IS 405
      LOCATION MODE IS          CALC USING ORD-NUMBER
                                DUPLICATES NOT ALLOWED
                                WITHIN ORD—AREA.
      03  ORD—NUMBER            PIC X(4).
      03  ORD—DATE              PIC X(6).

      RECORD NAME IS ORDERED-ITEM
                                RECORD ID IS 410
      LOCATION MODE IS VIA ITEMS—IN—ORDER.
```

```
   03  ITEM—NUMBER        PIC X(9).
   03  QUANTITY           PIC 9(6)V99.
   03  PRICE              PIC 9(7)V99.
   03  DELIVERY—DATE      PIC X(6).
```

```
SET DESCRIPTION.
  SET NAME IS ORDERS—OF—CUSTOMER                              שם החבורה
  ORDER IS SORTED
  MODE IS CHAIN LINKED TO PRIOR

  OWNER IS CUSTOMER NEXT DBKEY IS 1                           מיקום המצביע
                          PRIOR DBKEY IS 2                      ברשומה
 MEMBER IS ORDER NEXT DBKEY IS 1
                          PRIOR DBKEY IS 2
     ASCENDING KEY IS ORD-NUMBER                              מפתח המיון
     DUPLICATES ARE NOT ALLOWED
     OPTIONAL MANUAL.
   SET NAME IS ITEMS—IN—ORDER
   ORDER IS FIRST
   MODE IS CHAIN
   OWNER IS ORDER NEXT DBKEY IS 3                             רשומת אב
 MEMBER IS ORDERED—ITEM NEXT DBKEY IS 1                       רשומת בן
   MANDATORY AUTOMATIC.
```

מן הראוי לשים לב לדמיון שבין שפת הגדרת הסכימה לבין שפת התכנות COBOL. גם שפות ההגדרה האחרות של IDMS מזכירות בצורת הכתיבה את COBOL.

5.2 שפה להגדרת אמצעי האחסון

המשתמש קובע מספר מאפיינים פיסיים של בסיס הנתונים באמצעות שפה הקרוייה DMCL — Device Media Control Language. שפה זו משמשת לקביעת המיפוי בין הדפים הלוגיים לבין הגושים הפיסיים בקובץ, גודל מאגרי קלט/פלט והשטחים שבהם משתמשים.

```
DEVICE MEDIA DESCRIPTION.
  DEVICE MEDIA NAME IS CUST—DMCL
    OF SCHEMA NAME CUST—ORDERS.
BUFFER SECTION
  BUFFER NAME IS CUST—BUFF
  PAGE CONTAINING 1180 CHARACTERS
  BUFFER CONTAINS 5 PAGES.
```

```
AREA SECTION.
COPY CUST—AREA AREA.
COPY ORD—AREA AREA.
```

5.3 שפה להגדרת תת־סכימות

תוכניות היישום פונות אל בסיס הנתונים באמצעות תת־סכימה אשר מגדירה את התצפית של המשתמש על הסכימה. ניתן להגדיר מספר בלתי מוגבל של תת־סכימות המתייחסות לסכימה אחת.

תת־הסכימה מורכבת משני קטעים: קטע אחד המזהה את תת־הסכימה וקטע שני אשר מגדיר את הנתונים שתוכנית היישום תשתמש בהם. במסגרת זיהוי תת־הסכימה נקבע השם שלה ושם הסכימה שאליה היא מתייחסת והשם שניתן למתקן האחסון בהגדרת DMCL. בהגדרת הנתונים על המשתמש לציין את כל הרשומות, החבורות והשטחים שאליהם ברצונו לגשת. ניתן להרשות לתוכנית היישום לראות את כל הרשומה או רק חלק מהשדות המרכיבים אותה.

במסגרת תת־הסכימה ניתן להגדיר גם את פעולות העדכון שתוכנית היישום תורשה לבצע בבסיס הנתונים. אפשר לקבוע עבור כל פעולה אם היא מותרת או אסורה לתוכנית היישום.

דוגמא להגדרת תת־סכימה:

```
SUBSCHEMA IDENTIFICATION DIVISION.
SUBSCHEMA NAME IS CUST—REPORT OF SCHEMA NAME CUST—ORDERS.
   DEVICE MEDIA NAME IS CUST—DMCL.
SUBSCHEMA DATA DIVISION.
   AREA SECTION.
   COPY CUST—AREA AREA PRIVACY LOCK FOR UPDATE IS "NO".
   COPY ORD—AREA AREA PRIVACY LOCK FOR UPDATE IS "NO".
   RECORD SECTION.
      01 CUSTOMER.
         03 CUST—ID.
         03 CUST—ADDRESS.
      COPY ORDER RECORD.
   SET SECTION.
   COPY ORDERS—OF—CUSTOMER SET.
```

6. שפה לטיפול בנתונים

השפה לטיפול בנתונים מאפשרת לתוכנית מגוון של פעולות בבסיס הנתונים. תוכנית היישום הנטענת לזכרון המחשב מחולקת לשני חלקים:

— **הוראות התוכנית** לעיבוד הנתונים,

— **שטח עבודה** (UWA — User Working Area) המכיל את הרשומות שהתוכנית מעבדת.

שטח זה מכיל גם שטח תקשורת מיוחד בין תוכנית היישום לבין מערכת IDMS, שגודלו נקבע ל־200 בתים. אל שטח התקשורת מעבירה המערכת מידע עבור תוכנית היישום בהתאם לפעולות שבוצעו, כמו לדוגמא **קוד מצב** (Status) המציין אם הבקשה של תוכנית היישום בוצעה בהצלחה או לא. השדות המוגדרים בשטח התקשורת הם:

PROGRAM-NAME:	שם תוכנית היישום,
ERROR-STATUS:	בשדה זה יוצג קוד המצב של פקודת DML האחרונה שבוצעה,
DBKEY:	מפתח הזיהוי של הרשומה האחרונה שתוכנית היישום טיפלה בה,
RECORD-NAME:	שם הרשומה האחרונה שטופלה על ידי תוכנית היישום,
AREA-NAME:	שם השטח האחרון שתוכנית היישום ניגשה אליו,
ERROR-SET:	שם החבורה שהתוכנית טיפלה כאשר קרתה שגיאה,
ERROR-RECORD:	שם הרשומה שהתוכנית טיפלה בה כאשר קרתה שגיאה,
ERROR-AREA:	שם השטח שהתוכנית טיפלה כאשר קרתה שגיאה,
DIRECT-DBK:	שם השדה המשמש לשיטת איתור DIRECT.

כדי שהתוכנית היישום תוכל למקם את עצמה בבסיס הנתונים מנהל IDMS **מראי מקום** (Currency Indicators) המכילים את **מפתחות הזיהוי** (DB-Keys) שתוכנית היישום טיפלה בהן לאחרונה. מראי המקום המנוהלים על ידי IDMS הם:

RUN-UNIT:	המפתח של הרשומה האחרונה שתוכנית היישום טיפלה בה,
RECORD-TYPE:	לכל סוג רשומה בתת־סכימה קיים מראה מקום אחד המכיל את המפתח של הרשומה האחרונה שתוכנית היישום טיפלה בה,
SET:	לכל חבורה בתת־סכימה קיים מראה מקום אחד המכיל את המפתח של רשומת האב האחרונה שתוכנית היישום טיפלה בה,
AREA:	לכל שטח בתת־סכימה קיים מראה מקום אחד המכיל את המפתח של הרשומה האחרונה שתוכנית היישום טיפלה בה,

ניתן לחלק את פקודות DML של מערכת IDMS לשלש קבוצות: פקודות בקרה, פקודות אחזור ופקודות עדכון, אשר נסקור בהמשך.

6.1 פקודות בקרה

פקודות הבקרה משמשות לניהול השטחים השונים המטופלים על־ידי תכנית היישום במהלך העיבוד.

BIND: פקודה זו קושרת את שטחי התקשורת של IDMS ואת הרשומות שהתוכנית מעבדת עם שטחי העבודה (UWA). לפני שניתן לבצע עיבוד כלשהו של בסיס הנתונים חייב להתבצע קישור של תת־הסכימה באמצעות פקודת BIND.

READY: פקודה זו מכינה את השטח לעיבוד על־ידי תוכנית היישום. אין אפשרות לעבד שטח שלא "נפתח" באמצעות פקודה זו לשם קריאה בלבד או גם לשם עדכון. אם מספר תוכניות רצות בו־זמנית ומעדכנות את אותו השטח, ניתן למנוע עדכון בו זמני על־ידי פתיחה עם אופצית PROTECT. אם רוצים למנוע גם את קריאת השטח על־ידי תוכניות יישום אחרות ניתן להשתמש באופציה EXCLUSIVE.

FINISH: פקודה זו מבצעת את סגירת השטח בגמר העיבוד.

6.2 פקודות אחזור

פקודות אלה משמשות לאחזור רשומות מתוך בסיס הנתונים והבאתן לשטח העבודה (UWA).

FIND/GET: פקודת FIND מאפשרת איתור מופע של רשומה בבסיס הנתונים. הרשומה הופכת להיות הרשומה האחרונה ומראי המקום מעודכנים בהתאם. פקודת GET מעבירה את הרשומה אל שטח העבודה. ניתן לשלב את פקודות FIND ו־GET לפקודה יחידה — OBTAIN, המאתרת ומעבירה את הרשומה. קיימים מבנים שונים לפקודת FIND/OBTAIN:

א. **CALC**: פקודת FIND/OBTAIN CALC מאחזרת רשומה על פי המפתח הלוגי שלה אשר מועבר לשטח העבודה לפני ביצוע הפקודה. לדוגמא,

```
MOVE 104 TO CUST—ID.
FIND CALC CUSTOMER.
```

אם קיימות בבסיס הנתונים רשומות אחדות בעלות מפתח זהה, תאוחזר רק הרשומה הראשונה מביניהן. שאר הרשומות תאוחזרנה באמצעות פקודות FIND NEXT DUPLICATE:

```
FIND NEXT DUPLICATE CUSTOMER.
```

ב. **WITHIN SET**: לאחר שהתוכנית מאתרת רשומה כלשהי ניתן להתחיל בניווט בבסיס הנתונים על פי החבורות השונות שהוגדרו. פקודת FIND/OBTAIN WITHIN SET מאפשרת את האופציות הבאות:

FIRST — רשומת בן ראשונה בחבורה,
LAST — רשומת בן אחרונה בחבורה,
NEXT — רשומת הבן הבאה בהתייחס לרשומה האחרונה שטופלה,
PRIOR — רשומת הבן הקודמת בהתייחס לרשומה האחרונה שטופלה,
OWNER — רשומת האב של החבורה,

USING — לאיתור רשומה בחבורה הממויינת לפי מפתח כלשהו,
N-th — לאיתור הרשומה ה־N־ית מהרשומה האחרונה שטופלה.

דוגמא לפקודות אחזור:

```
MOVE 104 TO CUST—ID.
OBTAIN CALC CUSTOMER.
OBTAIN FIRST ORDER WITHIN ORDERS—OF—CUSTOMER.
— — —
OBTAIN NEXT ORDER WITHIN ORDERS—OF—CUSTOMER.
```

מכיון שרשומות הבן בחבורה ORDERS—OF—CUSTOMER ממויינות לפי מספר הזמנה ניתן לגשת לרשומות ההזמנה על פי המפתח.

```
MOVE 104 TO CUST—ID.
MOVE 805 TO ORD—NUMBER.
FIND CALC CUSTOMER.
OBTAIN  ORDER WITHIN ORDERS—OF—CUSTOMER
        USING ORD—NUMBER.
```

ג. **WITHIN AREA**: אפשר לנווט בבסיס הנתונים על פי המיקום הפיסי של תוכנית היישום. פקודת FIND/OBTAIN WITHIN AREA מאפשרת את האופציות הבאות:

FIRST — רשומה ראשונה בשטח,
LAST — רשומה אחרונה בשטח,
NEXT — הרשומה הבאה בשטח בהתייחס לרשומה האחרונה שטופלה,
PRIOR — הרשומה הקודמת בשטח בהתייחס לרשומה האחרונה שטופלה,
N—th — הרשומה הN—ית מהרשומה האחרונה שטופלה.

```
OBTAIN FIRST WITHIN CUST—AREA
— — — —
OBTAIN NEXT WITHIN CUST—AREA.
```

אם באותו שטח מאוחסנות מספר סוגי רשומות, ניתן לשלוף סוג רשומה מסויים כאשר מציינים את שם הרשומה.

```
OBTAIN NEXT ORDERED—ITEM WITHIN ORD—AREA.
```

ד. **DB-KEY**: אם תוכנית היישום מכירה את מפתח הזיהוי ניתן לגשת ישירות לרשומה באמצעות פקודת FIND/OBTAIN DB-KEY. את ערך מפתח הזיהוי אפשר לקבל באמצעות הפקודה ACCEPT. על המשתמש להגדיר שדה בינרי, PIC S9(8) COMP, שישמש לאחסון המפתח. סדר הפעולות:

```
MOVE 104 TO CUST—ID
OBTAIN CALC CUSTOMER.
OBTAIN NEXT ORDER WITHIN ORDERS—OF—CUSTOMER.
 ACCEPT KEY—1 FROM ORDERS—OF—CUSTOMER CURRENCY
— — — —
OBTAIN DB—KEY IS KEY—1.
```

שדה בינרי שהוגדר (← KEY—1)

גישה ישירה לפי המפתח (← OBTAIN DB—KEY IS KEY—1)

6.3 פקודות עדכון

פקודות אלה מאפשרות את עדכון הרשומות והקשרים בין הרשומות שבבסיס הנתונים.

STORE: הוספת רשומה חדשה לבסיס הנתונים מתוך שטח העבודה (UWA) שבתוכנית. על תוכנית היישום לעדכן את כל מראי המקום עבור החבורות שבהן משתתפת הרשומה בסוג חברות אוטומטית כדי שהמערכת תקשר את הרשומה החדשה לחבורות המתאימות.

DELETE: ביטול רשומה המופיעה בבסיס הנתונים. לפקודה זו יש מספר תבניות שונות:
DELETE ALL מבטלת את הרשומה ואת כל רשומות הבן הקשורות אליה.
DELETE PERMANENT מבטלת את הרשומה ואת כל רשומות הבן הקשורות **בקשר הכרחי** (Mandatory). כל רשומות הבן הקשורות בקשר אופציונלי מנותקות מהחבורה אך אינן מתבטלות.

DELETE SELECTIVE מבטלת את הרשומה ואת כל רשומות הבן הקשורות בקשר הכרחי ואת רשומות הבן הקשורות בקשר אופציונלי ואינן קשורות לכל חבורה אחרת.

MODIFY: פקודה לעדכון תוכן רשומה שמופיעה בבסיס הנתונים. אסור לעדכן את שדות המפתח של הרשומה. "עדכון" שדה מפתח יכול להעשות על ידי ביטול רשומה והוספה מחדש עם שדה מפתח שונה.

CONNECT: אם רשומת הבן קשורה לחבורה **בקשר ידני** (Manual) ניתן לקשור אותה לחבורה באמצעות פקודה זו.

DISCONNECT: מנתקת רשומת בן מחברותה בחבורה שבה הוגדרה בסוג חברות ידנית.

IF: הפקודה מאפשרת לתוכנית היישום לבדוק אם החבורה מכילה רשומת בן. קיימות מספר אופציות לפקודה זו, ולדוגמא אחת מהן:

IF ORDERS-OF-CUSTOMER NOT EMPTY...

7. סיכום

IDMS הינה מערכת לניהול בסיסי נתונים המבוססת על המלצות CODASYL DBTG. היא ומיישמת את רוב ההמלצות שבדוח וכוללת הרחבות נוספות. השפות DDL, DML הן שפות דמויות אנגלית ולכן קלות ללימוד ולהבנה. המשתמש אינו צריך להגדיר בצורה מפורשת את שטחי קלט/פלט שבתוכנית היישום. שטחים אלו מוגדרים בנפרד באמצעות DMCL. מערכת IDMS מאפשרת ייצוג מגוון מבנים לוגיים החל ממבנים פשוטים יחסית עד למבנים מורכבים מאד. DML היא שפה פרוצדורלית אשר מאפשרת את הניווט בבסיס הנתונים על בסיס **רשומה אחת בכל פעולה** (One-Record-at-a-Time). תכונה זו עלולה לגרום למורכבות ולקושי מיותרים בתוכנית היישום.

למערכת מצורפות תוכניות שירות אחדות המסייעות באחזקת בסיס הנתונים. לאחרונה הוכרז על מישק טבלאי ל-IDMS, אשר מאפשר להציג למשתמש מבנים של רשומות שטוחות.

פרק יג

TOTAL

1. מבוא
2. מושגי יסוד
3. מבנים פיסיים
4. מבנים לוגיים של נתונים
5. שפה להגדרת נתונים
6. שפה לטיפול בנתונים
7. השוואה בין מערכת TOTAL לבין המלצות CODASYL
8. סיכום

1. מבוא

מערכת TOTAL פותחה על ידי חברת .Cincom Systems Inc ונמצאת בשימוש במספר רב של מחשבים גדולים וקטנים (במערכות IBM, NCR, CDC, ועוד). המבנה הבסיסי ב־TOTAL הינו מבנה היררכי בעל שתי רמות אשר מאפשר ליצור מבני נתונים מורכבים וגם להציג רשתות. המערכת מאפשרת בניית מאגר נתונים מרכזי לשירות כל תוכניות היישומים. כל אחד מהגורמים המשתמשים תורם את חלקו בבניית מאגר הנתונים ונוטל ממנו את הדרוש לו. המערכת מאפשרת אינטגרציה של בסיסי הנתונים של היישומים השונים למאגר נתונים אחד. המונחים שמשתמשים בהם במערכת זו שונים מאלו המופיעים בדוח CODASYL מכיון שמערכת TOTAL פותחה לפני הופעת הדוח.

לאחרונה הוכרזה מערכת TIS אשר מכילה מישק טבלאי לבסיס הנתונים ומאפשרת הצגת רשומות שטוחות. השדות ברשומה השטוחה יכולים לבוא ממספר רב של קבצים פיסיים. הניווט בבסיס הנתונים, כדי לקבל את כל השדות של הרשומה השטוחה, מתבצע אוטומטית על־ידי מערכת TIS בהתאם להגדרה חד־פעמית של ה־DBA ומאוחסנת במילון המרכזי של המערכת (Dictionary).

2. מושגי יסוד

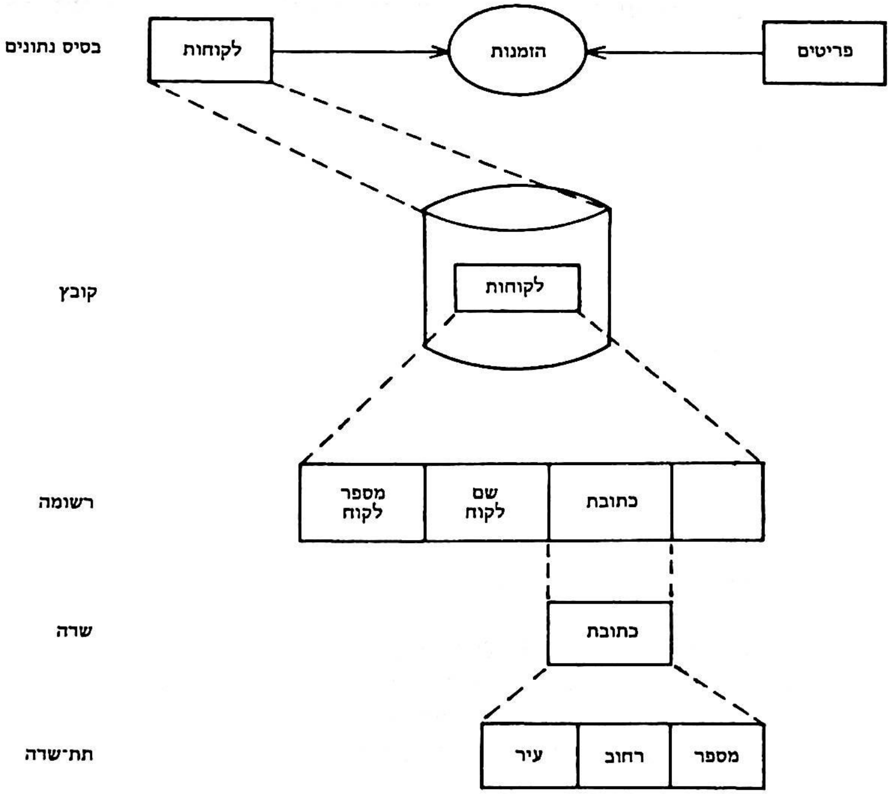
בתחילה נציג את המונחים המקובלים במערכות TOTAL:

— **שדה** (Data Field, Data Element): יחידת הנתונים הבסיסית במערכת. ההתייחסות לשדות היא לפי שמם ולא באמצעות כתובות פיסיות. שדה יכול להתחלק **לתת־שדות** (Sub-elements). לדוגמא השדה "תאריך" מתחלק לתת־שדות הבאים: יום, חודש, שנה.

— **רשומה** (Data Record): אוסף של מספר שדות בעלי משמעות לוגית מבחינת המשתמש. הרשומה הינה יחידת התקשורת הבסיסית בין בסיס הנתונים לבין תוכנית היישום. ניתן להגדיר מספר בלתי מוגבל של סוגי רשומות שונות.

— **קובץ** (Data Set): אוסף כל המופעים של הרשומות מאותו סוג. כל קובץ יכול להכיל מספר בלתי מוגבל של מופעים של רשומות.

— **בסיס נתונים** (Data Base): אוסף של כל הקבצים השונים.



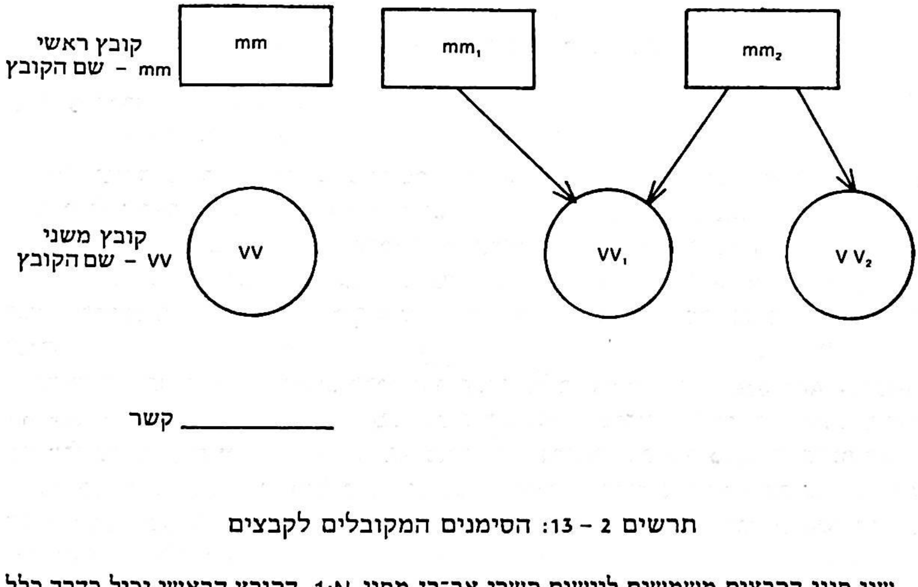
תרשים 1 – 13: מושגי היסוד ב־TOTAL

3. מבנים פיסיים

ב־TOTAL מבחינים בין שני סוגי קבצים:

— **קובץ ראשי** (Master file): קובץ בגישה ישירה שאינו תלוי בקובץ אחר. לכל רשומה בקובץ יש מפתח לוגי אחד המשמש את המערכת הן לאחסון הרשומה בבסיס הנתונים והן לאחזור הרשומה. הכתובת הפיסית של הרשומה נקבעת על ידי הפעלת תהליך חישובי כלשהו על התוכן של המפתח. לכל הרשומות בקובץ ראשי מבנה קבוע אחד.

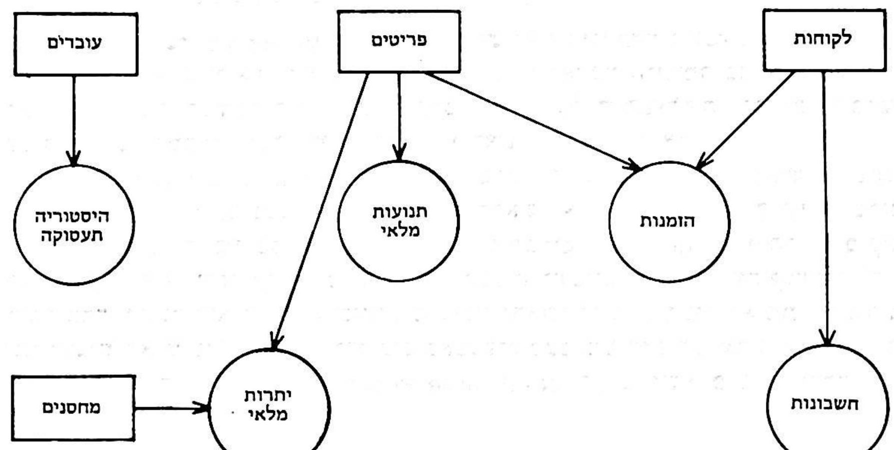
— **קובץ משני** (Variable file): קובץ התלוי בקובץ ראשי אחד או יותר. הגישה לרשומות המופיעות בקובץ המשני נעשית רק דרך הקובץ הראשי. אין אפשרות לגשת לרשומות בקובץ משני בגישה ישירה. בין שני סוגי הקבצים קיימים **קשרים** (Linkage Paths) הקושרים קובץ ראשי עם אחד או יותר קבצים משניים. שני קבצים ראשיים יכולים להיות קשורים לקובץ משני אחד. הקשר הוא תמיד חד־כיווני, מהקובץ הראשי לקובץ המשני. לא ניתן לקשור בין שני קבצים מאותו סוג. הקשרים מבוצעים באמצעות מצביעים המכילים את הכתובת היחסית של הרשומה בקובץ (RRN—Relative Record Number). בתרשימים נציין קובץ ראשי במלבן וקובץ משני בעיגול.



תרשים 2 - 13: הסימנים המקובלים לקבצים

שני סוגי הקבצים משמשים ליישום קשרי אב־בן מסוג 1:N. הקובץ הראשי יכיל בדרך כלל את הנתונים הקבועים או היציבים והקובץ המשני יכיל בדרך כלל את הנתונים המשתנים (דינמיים) בארגון. דוגמא לקבצים ראשיים: קובץ עובדים, קובץ פריטים, קובץ לקוחות. דוגמא לקבצים משניים: קובץ פרטי תעסוקה של העובד, קובץ תנועות מלאי של פריט, קובץ הזמנות של הלקוחות, קובץ חשבונות לתשלום וכדומה.

תרשים 3—13 מציג מבנה נתונים מורכב יחסית שניתן לבנות במערכות TOTAL.



תרשים 3 – 13: מבנה נתונים מורכב

3.1 מבנה קובץ ראשי

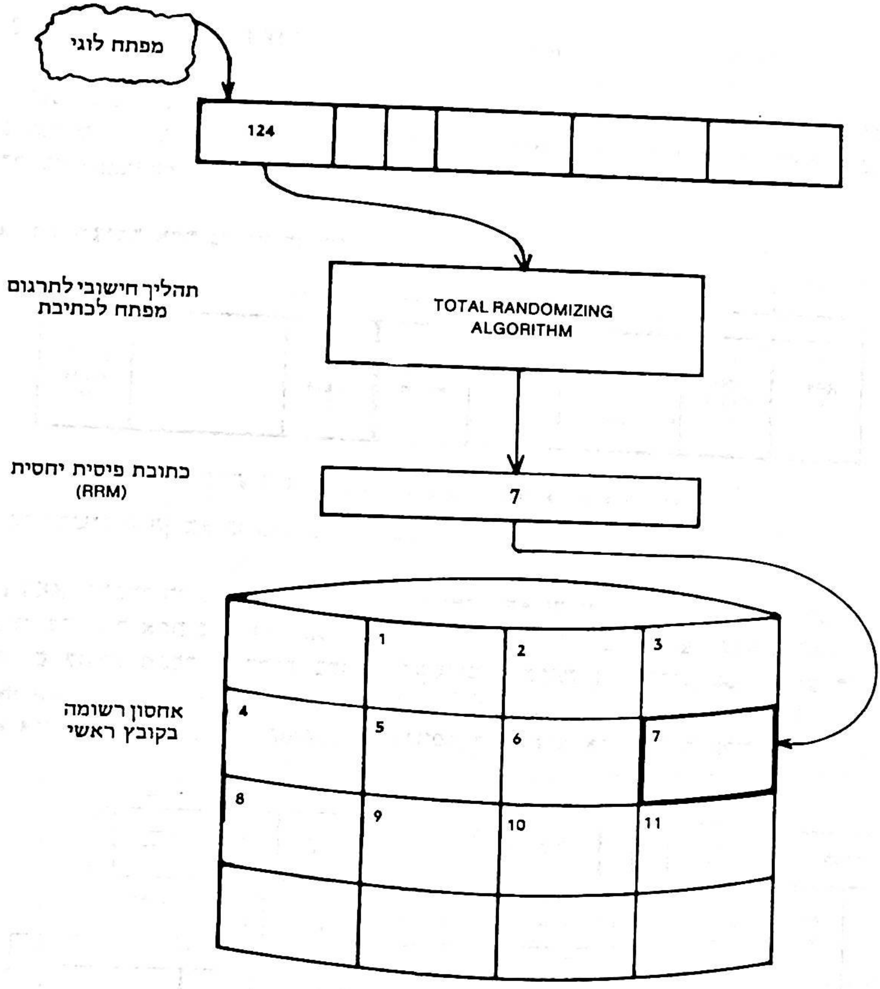
לכל רשומה בקובץ ראשי יש **שדה בקרה** (Control Field) אחד המשמש את המערכת כמפתח לוגי לאחסון ואחזור הרשומה. הרשומות בקובץ הראשי משמשות למעשה כנקודות כניסה לבסיס הנתונים ולכן אסור השימוש במפתחות כפולים בקובץ ראשי. ניתן ליישם קובץ ראשי כקובץ בגישה ישירה (BDAM) בלבד. אם בוחרים ב־BDAM או ב־VSAM-ESDS הגישה לרשומות היא בעזרת תהליך חישובי והתנהגות הקובץ היא כמו בקובץ רגיל בגישה ישירה.

תרגום המפתח לכתובת וחישוב הכתובת היחסית של הרשומה בקובץ (Relative Address Calculation) נעשים בשגרה אשר משתמשת בשיטות חישוביות שונות כדי לקבל מספר מינימלי של **כתובות זהות** (Synonyms). כאשר מתקבלת כתובת זהה מנסה המערכת לאתר כתובת פנוייה במקום קרוב ככל הניתן לכתובת המקורית. בדרך כלל הרשומה עם הכתובת הזהה תימצא באותו גוש פיסי ולכן ברוב המקרים הגישה לרשומה בקובץ הראשי תבוצע בפעולת קלט/פלט אחת.

לכל רשומה בקובץ הראשי יש מבנה קבוע אחד, כפי שמוצג בתרשים 5—13. מרכיבי הרשומה:

— **ROOT**: שדה המשמש את המערכת בלבד לשם שרשור הרשומות שתהליך החישוב קבע להן כתובת פיסית זהה. השדה משמש לקישור הרשומה עם הכתובת המקורית שלה.

— **CONTROL KEY**: שדה המפתח אשר נמצא בכל רשומה בקובץ הראשי. לכל רשומה שדה מפתח אחד בלבד.



תרשים 4 - 13: שיטת הגישה לקובץ ראשי

— **LINK PATH**: שדה זה קושר את הרשומות בקובץ הראשי לרשומות בקובץ המשני. לכל רשומה בקובץ הראשי יתכנו מספר רב של שדות קישור בהתאם למספר הקבצים המשניים הקשורים אל הקובץ הראשי. שדה הקישור מכיל את המצביע לרשומה הראשונה ולרשומה האחרונה בקובץ המשני הקשורות לרשומה בקובץ הראשי. אורך שדה הקישור 8 בתים.

— **DATA ITEM J**: שדות הנתונים של הרשומה.

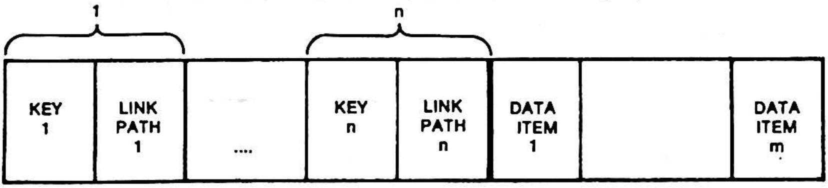
שורש	מפתח	קשר		קשר	שדה		שדה
ROOT	CONTROL KEY	LINK PATH 1	...	LINK PATH N	DATA ITEM 1	...	DATA ITEM M

תרשים 5 - 13: תבנית רשומה בקובץ ראשי

3.2 מבנה קובץ משני

קובץ משני יכול להיות בנוי כקובץ BDAM או כקובץ VSAM-ESDS. הגישה לרשומות בקובץ משני מבוצעת באמצעות המצביעים המופיעים ברשומות שבקובץ הראשי בשדות הקישור (Link Path). הרשומות בקובץ המשני יכולות להיות בשתי תבניות שונות.

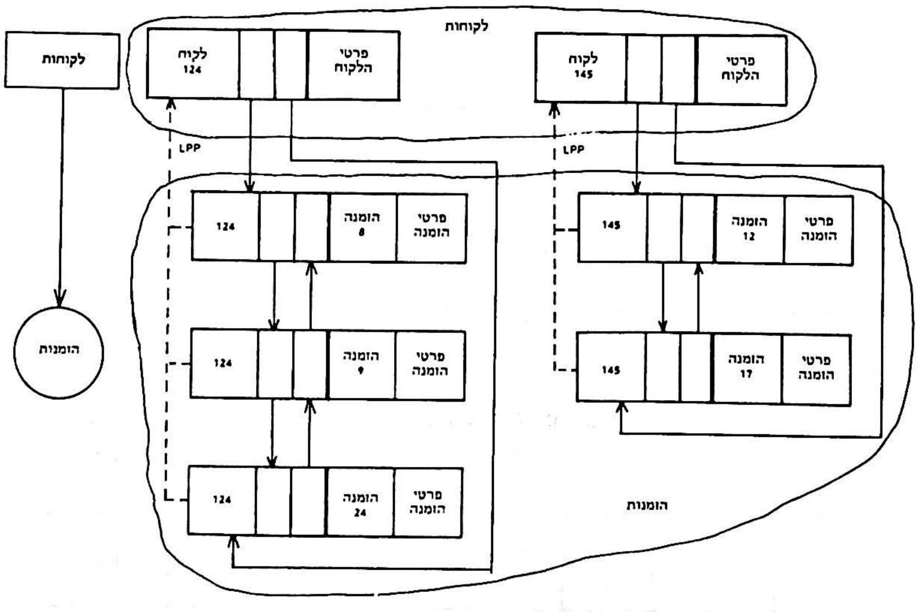
א. סוג רשומה אחד בקובץ המשני



תרשים 6 – 13: סוג רשומה אחד בקובץ המשני

כל הרשומות הן מאותו סוג, במבנה ובאורך זהה.

KEY i: המפתח הלוגי של הרשומה בקובץ הראשי שאליה קשורה הרשומה בקובץ המשני. מכיון שרשומה אחת בקובץ המשני יכולה להיות קשורה למספר רב של קבצים ראשיים עשויים להיות מספר מפתחות כמספר הקשרים. המפתח הזה נקרא גם **מצביע לוגי** (LPP—Logical Primary Pointer). והוא מקביל למצביע לרשומת האב במבנה CODASYL. הוא אינו מכיל את הכתובת הפיסית של רשומת האב אלא את המפתח הלוגי שלה.



תרשים 7 – 13: דוגמא לקשרים בין קובץ ראשי ומשני

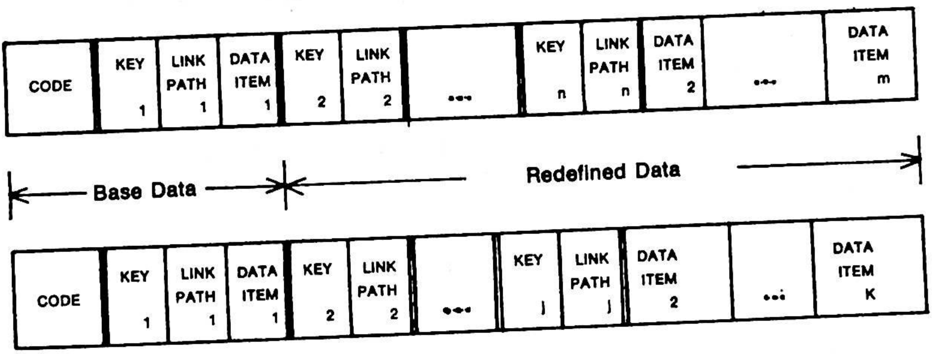
— **LINK PATH j**: אלו הם שדות הקישור שבין הרשומות בקובץ המשני הקשורות לאותה רשומה בקובץ הראשי. כל שדה קישור מכיל **מצביע לפנים**, לרשומה המשנית הבאה הקשורה לאותה רשומה בקובץ אב **ומצביע לאחור**, לרשומה המשנית הקודמת הקשורה לאותה רשומה בקובץ האב. אורך שדה הקישור 8 בתים.

— **DATA ITEM k**: שדות הנתונים של הרשומה המשנית.

בתרשים 7—13 מוצגת דוגמא של קשר בין קובץ ראשי לקוחות לבין קובץ משני הזמנות. המכיל רק סוג רשומה אחד. בדוגמא יש שני מופעים של רשומות בקובץ הראשי (לקוח מספר 124 ולקוח מספר 145). רשומת לקוח 124 קשורה לשלש רשומות הזמנה (הזמנות מספר 8, 9, 24).

ב. מספר סוגי רשומות בקובץ המשני (Coded Record)

ניתן להגדיר מספר תבניות שונות לרשומות המופיעות בקובץ המשני בתנאי שכל התבניות תהיינה באורך זהה. כל רשומה מכילה שני תווים המזהים את סוג הרשומה. הרשומות מחולקות לשני קטעים; **קטע בסיסי** הזהה לכל הרשומות בקובץ המשני **וקטע משתנה** בהתאם לסוג הרשומה. בתרשים 8—13 מוצגות תבניות של שני סוגי רשומות.



תרשים 8 - 13: שני סוגי רשומות בקובץ משני

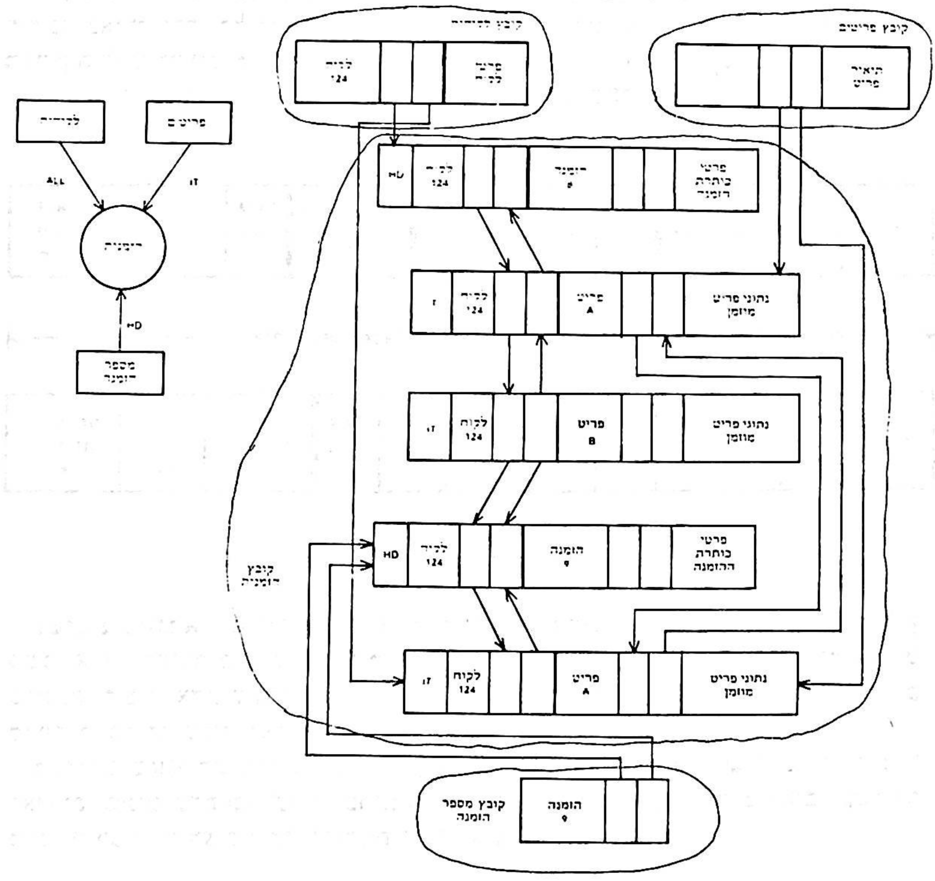
נשתמש בדוגמא שבתרשים 9—13 כדי להסביר את השימוש בשני סוגי רשומות בקובץ משני אחד: רשומת כותרת הזמנה המכילה פרטים כלליים על כל ההזמנה ורשומת פריט בהזמנה. הזמנה אחת עשויה להכיל מספר רב של פריטים מוזמנים ובהתאם לכך יהיה גם מספר הרשומות מסוג "פריט בהזמנה".

השימוש בסוגי רשומות שונים בקובץ המשני מאפשר גמישות בקישור שלהן לקבצים ראשיים שונים בהתאם לצורך. בדוגמא זו ניתן לקשור לקובץ ראשי פריטים. קשר זה מאפשר קבלת מידע עם כל ההזמנות שמופיע בהן פריט נתון.

רשומה אחת להזמנה	קוד יצרן	ערך הזמנה	תאריך הזמנה	LINK הזמנות	מספר הזמנה	LINK לקוח	מספר לקוח	סוג רשומה HD	כותרת הזמנה

רשומות אחדות להזמנה אחת	FILLER	כמות מוזמנת	LINK פריט	מספר פריט	LINK לקוח	מספר לקוח	סוג רשומה IT	פריט

תרשים 9 – 13: שני סוגי רשומות בקובץ משני הזמנות



תרשים 10 – 13: קשרים בין קבצים ראשיים וקובץ משני עם שני סוג רשומות

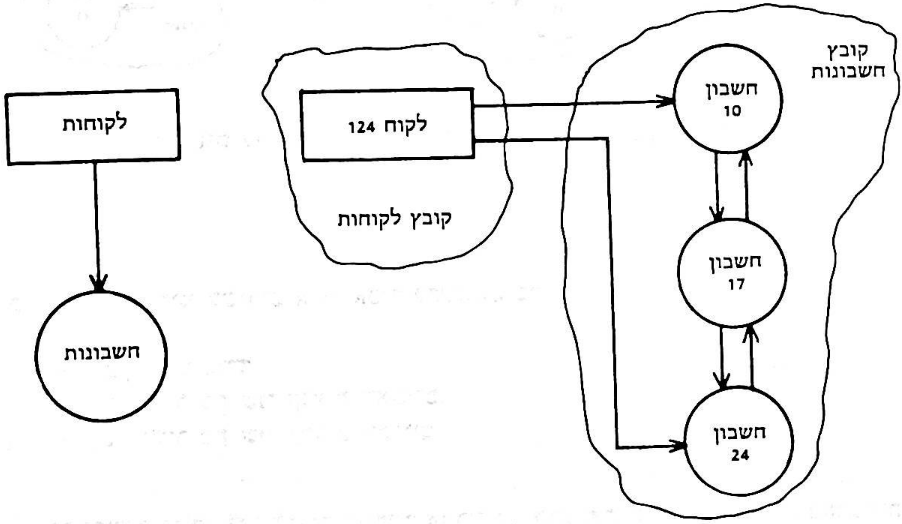
בתרשים 10—13 מוצגת דוגמא לשימוש בשני סוגי רשומות בקובץ המשני. ניתן לראות שלקוח 124 קשור לשתי הזמנות שונות, הזמנות מספר 8 ו־9. הזמנה מספר 8 מורכבת מפריטים A ו־B, והזמנה 9 מורכבת מפריט A בלבד. הרשומה של פריט A בקובץ ראשי פריטים קשורה לרשומות של פריט זה שבהזמנות 8 ו־9. הרשומה של הזמנה 9 בקובץ מספר הזמנה קשורה לכותרת ההזמנה בקובץ הזמנות.

3.3 סוגי קשרים בין קובץ ראשי לבין קובץ משני

בין קובץ ראשי לבין קובץ משני יכולים להיות קשרים בעלי אופי שונה בהתאם למספר הקבצים מסוג אחד הקשורים לקבצים מסוג שני. כפי שצויין קודם, אין אפשרות לקשור בין קבצים מאותו סוג, כלומר בין קבצים ראשיים לבין עצמם ובין קבצים משניים לבין עצמם.

— **קובץ ראשי בודד** (Single Master). הקובץ הראשי אינו קשור לקובץ משני כלשהו. קובץ ראשי בודד הוא בדרך כלל טבלת נתונים הבנוייה כקובץ ללא קשרים.

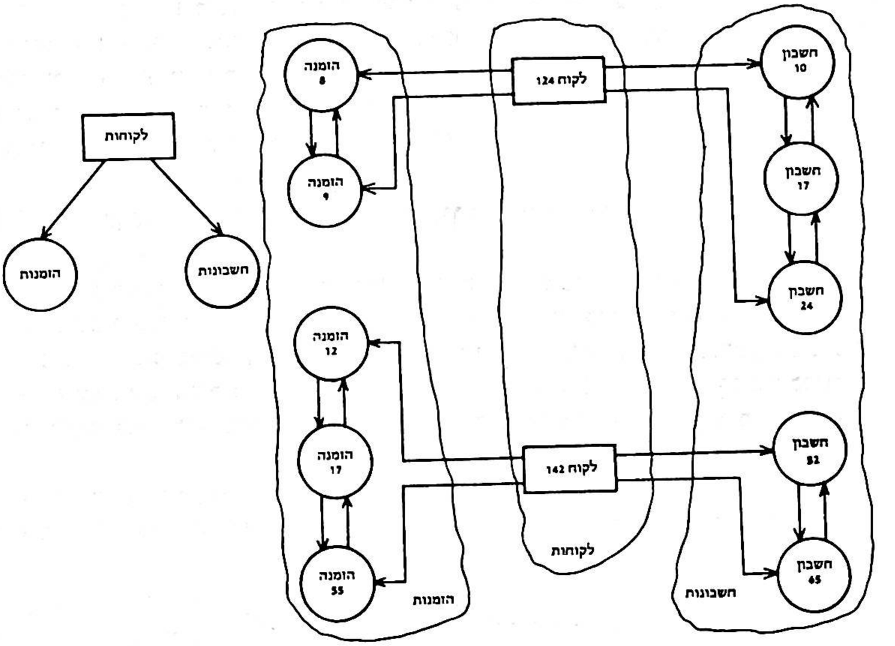
— **קובץ ראשי קשור לקובץ משני אחד** (Single Master — Single Variable). הקובץ הראשי קשור אך ורק לקובץ משני אחד, לדוגמא קובץ לקוחות הקשור לקובץ חשבונות.



תרשים 11 - 13: קובץ ראשי קשור לקובץ משני אחד

— **קובץ ראשי קשור למספר קבצים משניים** (Single Master — Multi Variable). הקובץ הראשי קשור למספר קבצים משניים, לדוגמא קובץ הלקוחות קשור לקובץ הזמנות ולקובץ חשבונות.

— **קובץ משני קשור למספר קבצים ראשיים** (Multi Master — Multi Variable). הקובץ המשני קשור למספר קבצים ראשיים שונים והקובץ הראשי קשור למספר קבצים משניים שונים. שיטה זו מאפשרת בניית רשתות נתונים במערכת TOTAL.

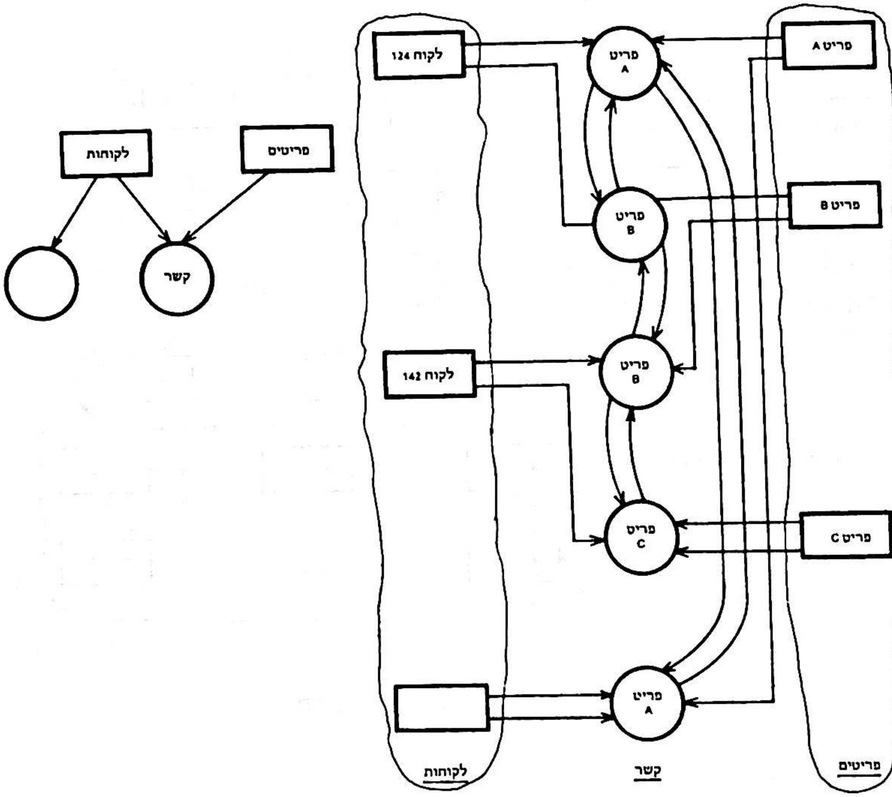


תרשים 12 – 13: קובץ ראשי קשור למספר קבצים משניים

קיימים מספר סוגי קשורים אשר אסור להשתמש בהם:

- קובץ משני בודד
- קשר ישיר בין שני קבצים ראשיים.
- קשר ישיר בין שני קבצים משניים.

אם דרוש קשר בין שני קבצים ראשיים או בין שני קבצים משניים יוצרים אותו באמצעות קשר אל קובץ משני משותף.

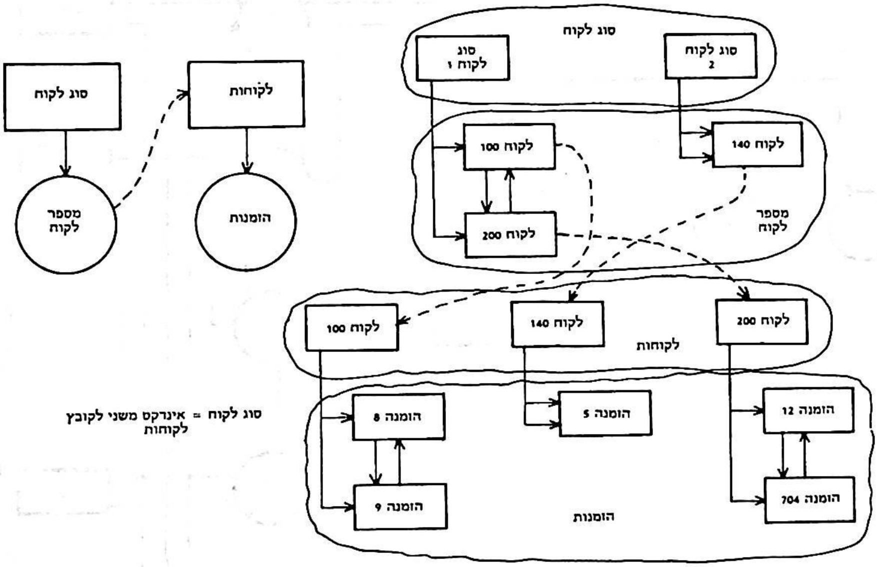


תרשים 13 – 13: קובץ משני קשור למספר קבצים ראשיים

3.4 אינדקס משני

מערכת TOTAL אינה תומכת בצורה ישירה באינדקס משני אך ניתן לתכנת את האינדקס המשני תוך שימוש בטכניקת הקבצים ובקשרים שביניהם. בתרשים 14—13 מוצגת דוגמא של אינדקס משני לקובץ לקוחות המאפשר גישה לרשומות הלקוחות לפי מפתח נוסף, סוג לקוח.

בקובץ הלקוחות מוצגות שלש רשומות של לקוחות: הלקוחות שמספרם 100 ו־200 מוגדרים "סוג 1" ולקוח 140 מוגדר "סוג לקוח 2". בונים קובץ ראשי "סוג־לקוח" המכיל את רשימת סוגי הלקוח האפשריים. לקובץ זה קשרים לקובץ משני "מספר־לקוח", המכיל את מספר הלקוחות הקשורים לכל סוג לקוח. רשומות הקובץ קשורות בהתאמה לקובץ ראשי "לקוחות". על־ידי קשר עקיף זה מקבלים את הקשר בין סוג לקוח לרשומת הלקוח. הקשר אינו מנוהל באופן אוטומטי על־ידי המערכת, אלא נקבע בתכנות על־ידי המשתמש. לפיכך, כאשר יש שינוי בקובץ הלקוחות, כמו למשל הוספת לקוח חדש או ביטול לקוח קיים, יש לעדכן בהתאם את מסלול הקשר לקובץ "סוג־לקוח".



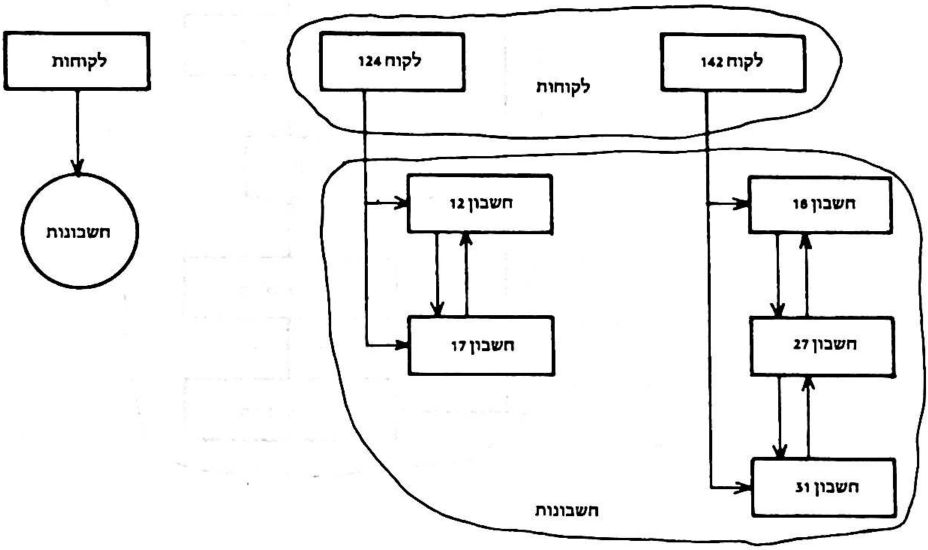
תרשים 14 – 13: אינדקס משני ב־TOTAL

4. מבנים לוגיים של נתונים

המבנים הבסיסיים של המערכת, הקובץ הראשי הקובץ המשני והקשר, מאפשרים להרכיב מבני נתונים מורכבים. גם מבנים אלו חייבים לקיים את הכללים אשר חלים על המבנים הבסיסיים. נסקור את המבנים המורכבים הנפוצים ביותר.

4.1 מבנה היררכי בעל רמה אחת (Single-level Hierarchy).

המבנה מוצג כקשר חד—רב־ערכי פשוט בעל יישום מידי. רשומה אחת בקובץ הראשי קשורה למספר רשומות בקובץ המשני, כפי שרשומת אב קשורה למספר רשומות בן במבנה הירכי.

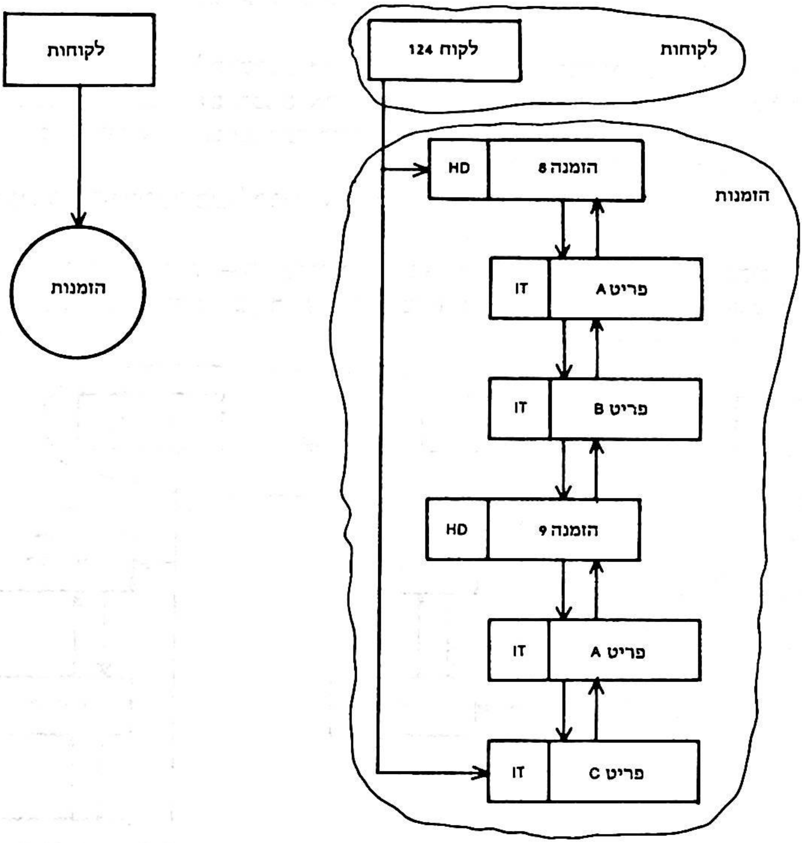


תרשים 15 – 13: מבנה היררכי פשוט

במבנה ההיררכי הפשוט המוצג בתרשים 15—13 לכל רשומת לקוח קשורות מספר רשומות חשבון בהתאם להזמנות שבוצעו. הגישה לרשומה בקובץ ראשי לקוחות הינה ישירה ומתבצעת על פי המפתח הלוגי שלה, לפי מספר הלקוח. הגישה לרשומות החשבון מתבצעת באמצעות המצביעים המרכיבים את השרשרת הדו־כיוונית בקובץ הראשי ובקובץ המשני.

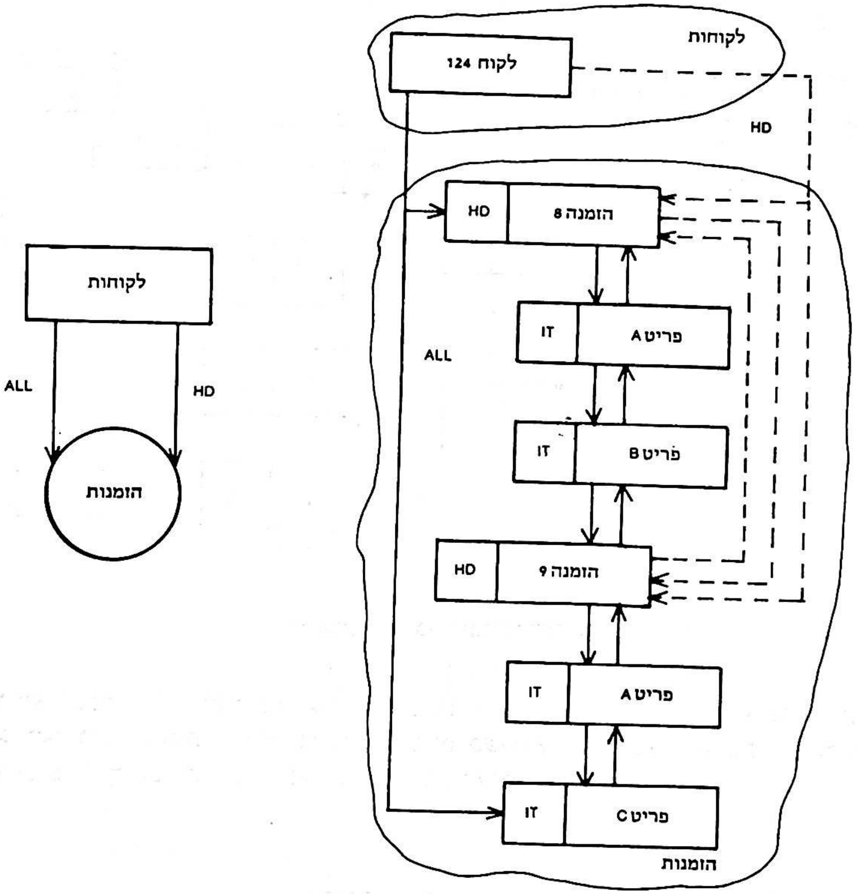
4.2 מבנה היררכי בעל מספר רמות (Multi-level Hierarchy).

כדי ליישם מבנה בעל רמות אחדות משתמשים במספר סוגי רשומות בקובץ המשני. אפשר לנצל תכונה נוספת של המערכת שהיא אפשרות הקשר בין רשומה בקובץ הראשי לרשומה כלשהי בקובץ המשני. במבנה ההיררכי שנוצר משמעות הדבר -אפשרות גישה ישירה לרשומות ברמה נמוכה יותר ללא מעבר דרך רשומת האב שלהן.



תרשים 16 – 13: מבנה הירכי בשתי רמות

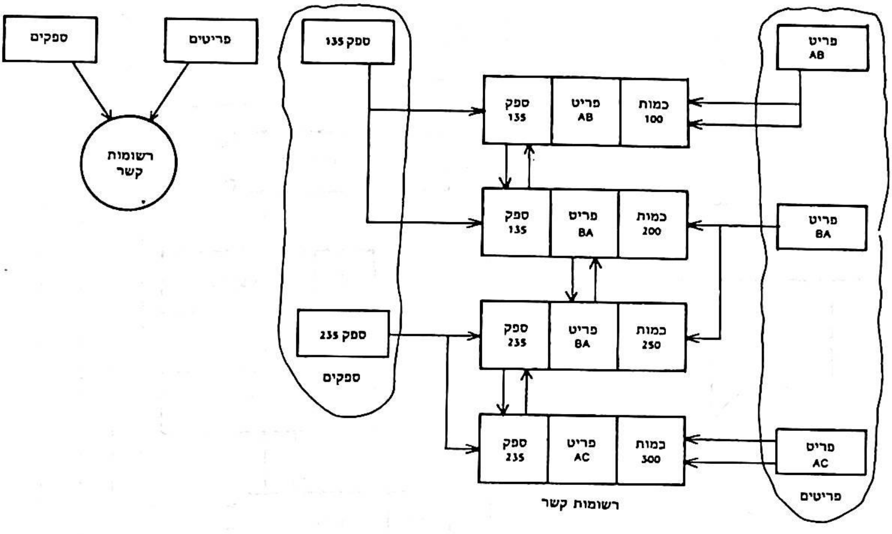
תרשים 16—13 מציג דוגמא ליישום המבנה ההירככי בשתי רמות. בשיטה זו כל הרשומות המשתתפות במבנה קשורות זו לזו בקשר דו־כיווני אחד. סדר הסריקה של המבנה הוא מרשומת כותרת בהזמנה הראשונה, לאחריה רשומות הפריטים שלה ולאחריהן רשומת כותרת ההזמנה הבאה וכן הלאה. בפועל, זוהי סריקה סדרתית של שרשרת המכילה שני סוג רשומות. כדי להגיע לכותרת של הזמנה מספר 9 למשל, יש לקרוא את כל הרשומות המופיעות לפניה. כדי לשפר את מהירות הגישה לרשומות משתמשים בקישור נוסף בין הקובץ הראשי לבין הקובץ המשני. הקשר השני קושר סוג רשומה אחד בלבד, למשל את כותרות ההזמנה. הקשר הזה יוצא שרשרת המכילה את רשומות הכותרת בלבד ועל ידי כך מאפשר גישה מהירה יותר אל רשומת כותרת מבוקשת.



תרשים 17 – 13: מבנה הירככי עם גישה מהירה לרשומות כותרת

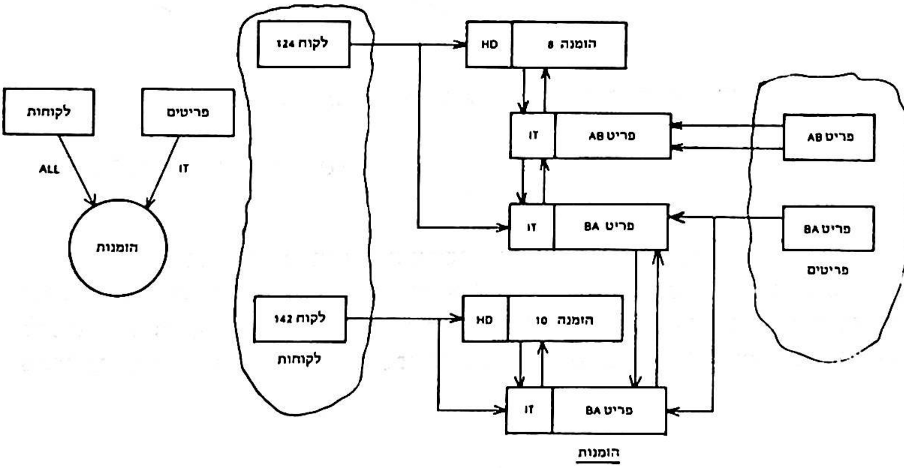
4.3 מבנה רשת (Network)

מבנה רשת אפשר ליישם באמצעות הקשר שבין קובץ משני אחד למספר קבצים ראשיים, וכדוגמא נשתמש בקשר רב—רב־ערכי אשר קיים בין ספקים לפריטים למשל. בדומה להמלצות CODASYL מוצג קשר זה על ידי שני קשרים מסוג חד—רב־ערכי, אשר כאן מקבל ביטוי בקישור קובץ משני אחד למספר קבצים ראשיים, כפי שמוצג בתרשים 18—13.



תרשים 18 – 13: מבנה רשת פשוט

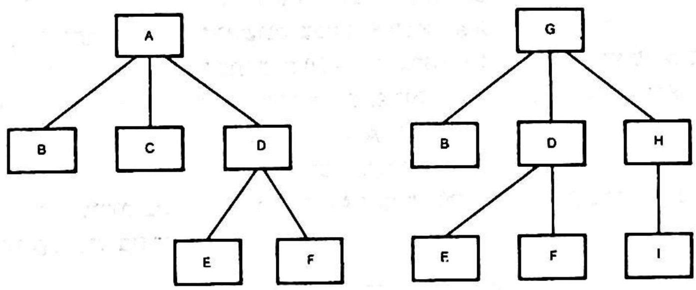
דוגמא נוספת למבנה רשתי מופיעה בתרשים 19—13 המתאר שימוש בקובץ משני עם שני סוג רשומות. רשומות הפריט בקובץ ההזמנות קשורות הן לקובץ הלקוחות והן לקובץ הפריטים ובכך יוצרות קשר רב—רב־ערכי בין הלקוחות לבין הפריטים.

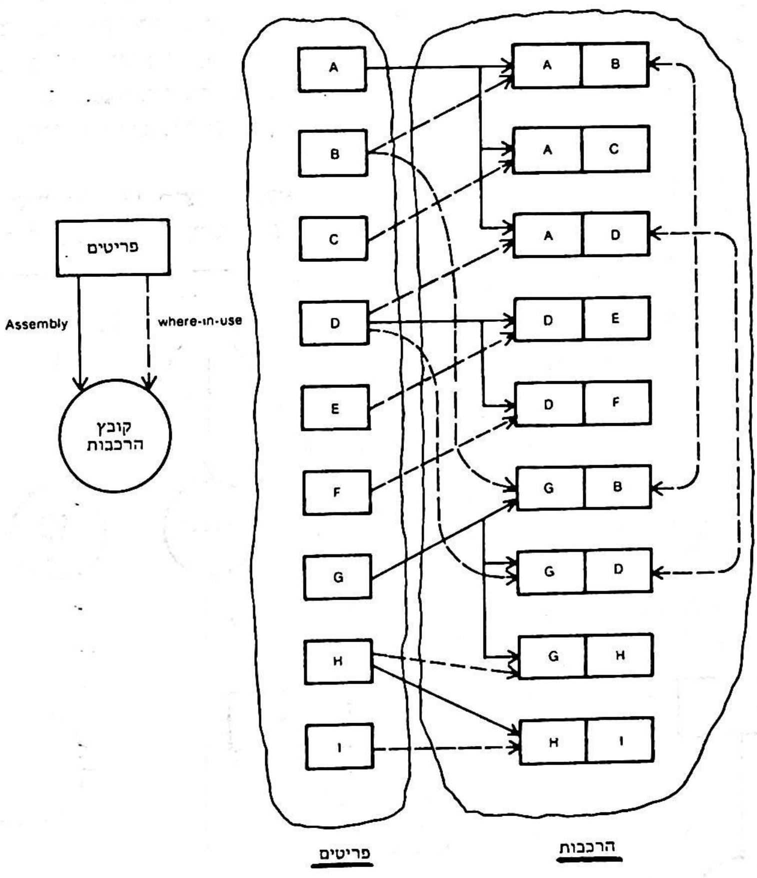


תרשים 19 – 13: מבנה רשת מורכב

4.4 מבנה עץ מוצר (Bill-of-Material Structure)

במערכת TOTAL לא ניתן להציג קשרים בין רשומות בקובץ ראשי. כדי לתאר את הקשרים שבין הרשומות של הפריטים המרכיבים את עץ המוצר יש להשתמש בטכניקה של





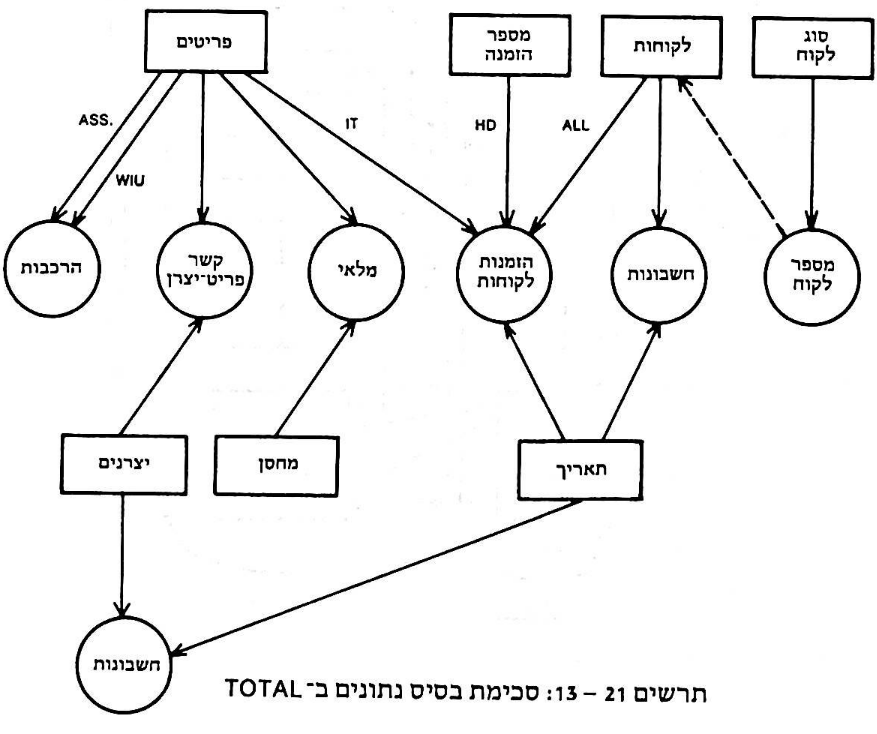
תרשים 20 – 13: מבנה עץ־מוצר

קובץ ראשי, קובץ משני וקשרים בין הרשומות שלהם. הקובץ הראשי מכיל את רשומות הפריטים. הקובץ המשני מכיל רשומות המתארות רמה של תת־הרכבה בלבד, כלומר יחס אב־בן, קשרים בין רשומות השייכות לתת־הרכבה אחת וקשרים בין רשומות שפריטים שלהן נמצאים [illegible]ית־הרכבות שונות. ברשומות שבקובץ הרכבות צריך להיות שדה המציין את מספר המופעים של הפריט בהרכבה. הקובץ הראשי קשור אל הקובץ המשני בשני סוג קשרים: קשר מהפריט אל הרשומה הראשונה של **תת־ההרכבה** (Assembly) שלו וקשר מהפריט אל הרשומה הראשונה בתת־ההרכבה הראשונה שבה הוא **כלול** (Where-In-Use).

[illegible] מפורט של עץ מוצר ניתן בתרשים 20—13. בדוגמא זו קיימים קובץ פריטים, קובץ הרכבות וקשרים. הרשומות של קובץ הרכבות מתארות קשרי הרכבות בלבד ולא פריטים בודדים. כך למשל הרשומות A—B ,A—C ,A—D מתארות תת־הרכבה אחת, והרשומות D—E ,D—F מתארות תת־הרכבה אחרת, אשר כלולה בתת־ההרכבה הראשונה. הרשומות בהרכבה A למשל קשורות ביניהן. רשומה A—B קשורה לרשומה G—B בתת־הרכבה אחרת מכיון שהפריט B מופיע בשתיהן.

הקשר מסוג Assembly שבין קובץ פריטים (ראשי) לבין קובץ הרכבות (משני) מוביל מרשומת פריט לכל הרשומות בהרכבה שלו. ובדוגמא, רשומה A קשורה אל רשומות A—B, A—B ,A—D; רשומה D קשורה לרשומות D—E, D—F וכך הלאה. קשר מסוג Where—In-Use מוביל מכל פריט אל רשומות הרכבה שבהן הוא משתתף. כך למשל מפריט D יהיו קשרים אל רשומות A—D, G—D ששייכות לשתי תת־הרכבות שונות.

בתרשים 21—13 מוצגת דוגמא לסכימה מורכבת על בסיס נתונים בארגון תעשייתי, לתיאור הסכימה יש להשתמש בכל סוגי הקשרים שנדונו בפרק זה.



תרשים 21 – 13: סכימת בסיס נתונים ב־TOTAL

5. שפה להגדרת נתונים

כדי להקים בסיס נתונים יש להגדיר תחילה את הסכימה שלו. ההגדרה כוללת את הקבצים, הרשומות והקשרים ביניהם ומספר תכונות פיסיות של הקבצים. במערכת TOTAL נעשות ההגדרות בשפה מיוחדת (DDL) העובדת הידור בתהליך הקרוי Data Base) DBGEN Generation).



תרשים 22 – 13: הידור הסכימה

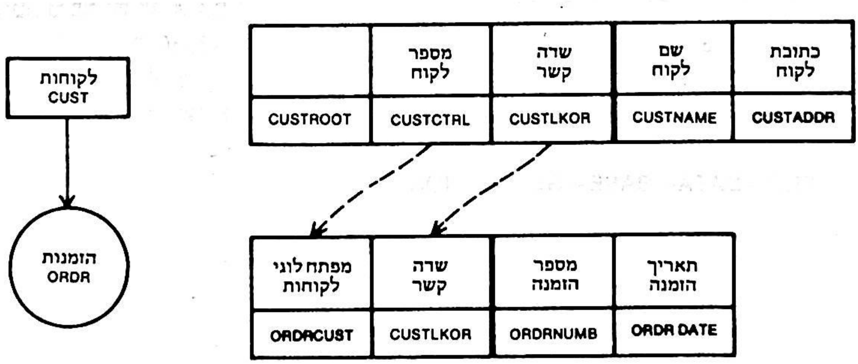
לכל קובץ במערכת ניתן שם מזהה חד־משמעי בן ארבעה תוים. שם זה משמש קידומת קבועה גם בשמות השדות המרכיבים את הרשומות בקובץ. לשדה המפתח יש מבנה מיוחד xxxxCTRL, שבו CTRL מציין את כעובדה שזהו **שדה מפתח** (Control Key) ובמקום xxxx רושמים את שם הקובץ. בכל רשומה יש להגדיר שדה ראשון בן 8 תווים בעל שם קבוע xxxxROOT המשמש את TOTAL בניהול המערכת.

ליצירת קשר בין שני קבצים יש להשתמש בכללים הבאים:

— בקובץ הראשי יוגדר שדה קשר בן 8 בתים על ידי השם xxxxLKYY כאשר YY הינו שם סימבלי הניתן על ידי המשתמש, ובמקום xxxx ירשם שם הקובץ.

— בקובץ המשני יהיו שני שדות לכל קשר: שדה המפתח הלוגי של הקובץ הראשי במבנה vvvvxxxx כאשר xxxx הוא שם הקובץ הראשי וְ־vvvv הוא שם הקובץ המשני. שדה הקשר השני יהיה בעל שם זהה לשדה הקשר בקובץ הראשי.

נשתמש בדוגמא פשוטה להגדרת סכימה של בסיס נתונים CUSTOMERS המכילה קובץ ראשי <u>לקוחות</u> CUST וקובץ משני <u>הזמנות</u> ORDR. שני הקבצים מוצגים בתרשים 23—13 ולאחריו — ההגדרה ב־DDL.



תרשים 23 – 13: תבנית שני קבצים להגדרת סכימה

	BEGIN—DATA—BASE—GENERATION:	
	DATA—BASE—NAME=CUSTOMERS.	שם בסיס הנתונים
	BEGIN—MASTER—DATA—SET:	תחילת הגדרה של קובץ ראשי
	DATA—SET—NAME=CUST.	שם הקובץ הראשי
	MASTER—DATA:	תיאור הקובץ הראשי
מבנה לוגי	CUSTROOT = 8	
	CUSTCTRL = 7	שדה המפתח (מספר לקוח)
	CUSTLKOR=8	שדה הקשר לקובץ משני
	CUSTNAME=40	
	CUSTADDR=30	
	END—DATA:	
מבנה פיסי	DEVICE=FBA	סוג הדיסק (גושים באורך קבוע—למשל יבמ 3370)
	ACCESS—METHOD=ESDS	קובץ VSAM/ESDS
	VSAM—CONTROL—INTERVAL—SIZE=4096	גודל הגוש
	TOTAL—LOGICAL—RECORDS=2000	מספר רשומות משוער בקובץ
	END—MASTER—DATA—SET:	
	BEGIN—VARIABLE—ENTRY—DATA—SET:	תחילת ההגדרה של קובץ משני
	DATA—SET—NAME=ORDR.	שם הקובץ המשני
	BASE—DATA:	
	ORDR CUST=7	המפתח הלוגי של הקובץ הראשי (מספר לקוח)
	CUSTLKOR=8=ORDRCUST	שדה קשר לקובץ הראשי
	ORDRNUMB=6	מספר הזמנה
	ORDRDATE=6	
	END—DATA:	
	ACCESS—METHOD=BDAM	שיטת גישה ישירה BDAM
	DEVICE=3330	סוג הדיסק (יבמ 3330)
	LOGICAL—RECORDS—PER—BLOCK=16	מספר רשומות לגוש
	TOTAL—LOGICAL—RECORDS=20000	מספר רשומות משוער בקובץ
	END—VARIABLE—ENTRY—DATA—SET:	
	END—DATA—BASE—GENERATION.	

אם הקובץ המשני מכיל סוגי רשומות אחדים מגדירים תחילה את השדות המשותפים ואחר כך את השטחים בהתאם לסוג רשומה. לדוגמא:

	BASE—DATA:	
	ORDRCODE=2	
משותף	ORDRCUST=7	
	CUST LKOR=8=ORDRCUST	שדה קשר
	ORDRNUMB=6	
HD	RECORD—CODE=HD	סוג הרשומה HD
	.1.IRDRDATE=6	
	.1.FILLER=19	"מילוי" כדי לקיים אורכי רשומה שווים
	RECORD—CODE=IT	סוג הרשומה IT
	.1.ORDRITEM=9	
IT	.1.ITEMLKOR=8=ORDRITEM	שדה קשר בין הרשומות בקובץ
	.1. ORDRQNTY=8	

6. שפה לטיפול בנתונים

השפה לטיפול בנתונים (DML—Data Management Language) מאפשרת גישה ישירה לרשומות ראשיות, גישה באמצעות מצביעים לרשומות משניות שבקובץ וגישה סדרתית בהתאם לסדר הפיסי של הרשומות. לקבלת השרותים של מערכת ניהול בסיס הנתונים TOTAL מתוך שפה מארחת (COBOL) יש לרשום הוראה במבנה הבא:

CALL DATBAS (operation, status, data-set-name, reference, linkage-path, control-field, element-list, record-area, END)

לכל סוג פעולה או הוראה יש להשתמש באוסף הפרמטרים המתאים, מתוך הרשימה שבמבנה ההוראה:

— OPERATION: הפקודה שיש לבצע, לדוגמא קריאה, כתיבה וכדומה.

— STATUS: שדה בתוכנית אשר יכיל בסיום הפעולה את קוד המצב אשר יציין אם הפעולה הושלמה כראוי או לא ומדוע.

— DATA—SET—NAME: שם הקובץ כפי שהוגדר בסכימה.

— REFERENCE: מספר המציין את המיקום היחסי של רשומה משנית מתחילת השרשרת. פרמטר זה מיועד להוראות בקבצים משניים בלבד.

— LINKAGE—PATH: שם הקשר כפי שהוגדר בסכימה. מכיון שרשומה משנית עשוייה להשתתף במספר קשרים יש לציין לפי איזה קשר יש לבצע את הפעולה.

— CONTROL—FIELD: המפתח של הרשומה בקובץ הראשי לצורך גישה ישירה. כאשר מטפלים ברשומות משניות שדה זה יכיל את המפתח הלוגי של הרשומה הראשית הקשורה לרשומה המשנית.

— ELEMENT—LIST: רשימת שמות השדות שיש לקרוא מהרשומה. השדות יקראו ויועברו לשטח העבודה בסדר שבו הם רשומים.

— RECORD—AREA: שטח עבודה בתוכנית שאליו יועברו השדות שיקראו מהרשומה או שאליו יועברו השדות שיכתבו אל הרשומה (עדכון).

הפקודות ב־TOTAL מתחלקות למספר קבוצות: פקודות לעיבוד סדרתי, פקודות לעיבוד קובץ ראשי. פקודות לעיבוד קובץ משני ופקודות בקרה.

6.1 פקודות לעיבוד סדרתי

RDNXT: פקודה זו מאפשרת קריאה של קובץ ראשי או קובץ משני בהתאם לסדר הפיסי של הרשומות. כל פקודה מחזירה רשומה אחת.

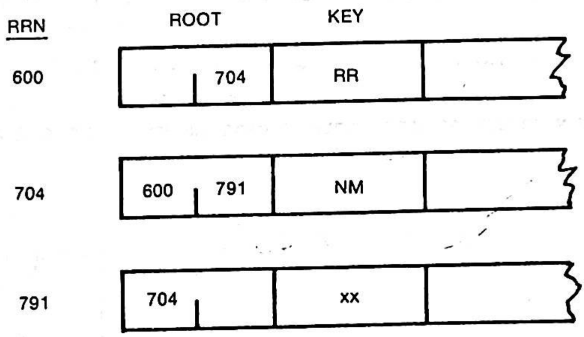
6.2 פקודות לעיבוד קובץ ראשי

READM: קריאת רשומה מקובץ ראשי לפי מפתח רשומה נתון.

WRITM: עדכון רשומה בקובץ הראשי אשר נקראה לפני־כן באמצעות פקודת READM.

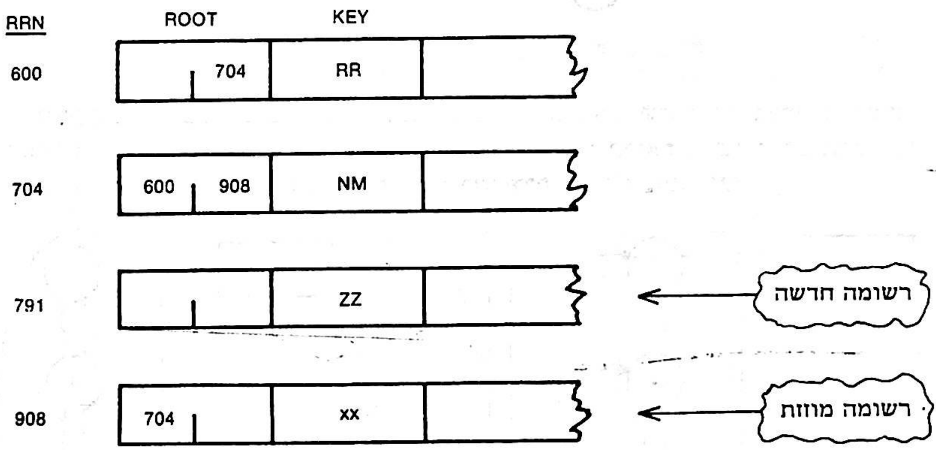
ADD—M: הוספת רשומה חדשה לקובץ ראשי. המערכת משתמשת במפתח ובתהליך חישובי לקביעת כתובת הרשומה בקובץ. אם המקום המיועד לרשומה תפוס על ידי רשומה אחרת, המערכת תבצע סריקה לאיתור מקום פנוי אחר כדי לכתוב אותה. בתרשים 24—13

ניתנת דוגמא לשלש רשומות שהיו צריכות לתפוס אותו מקום פיסי (Synonyms), ומוצג הקשר שנוצר ביניהן באמצעות המצביעים.



תרשים 24 – 13: תוכן שדה שורש עבור Synonyms

כאשר מגיעה רשומה חדשה אשר כתובתה המקורית היא 791, תוזז הרשומה עם מפתח xx למקום פנוי אחר והמצביעים יעודכנו בהתאם. ראה תרשים 25—13.

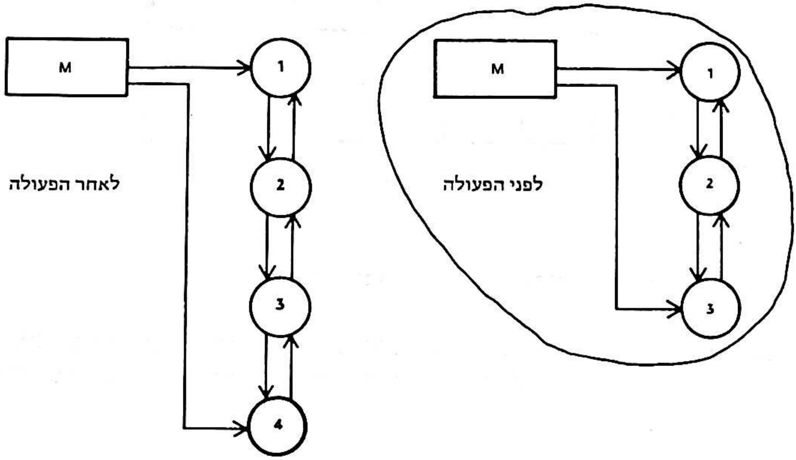


תרשים 25 – 13: כתיבת רשומה חדשה בקובץ

DEL—M: ביטול רשומה בקובץ ראשי. יש לבטל תחילה את כל הרשומות בקובץ המשני הקשורות לרשומה זו.

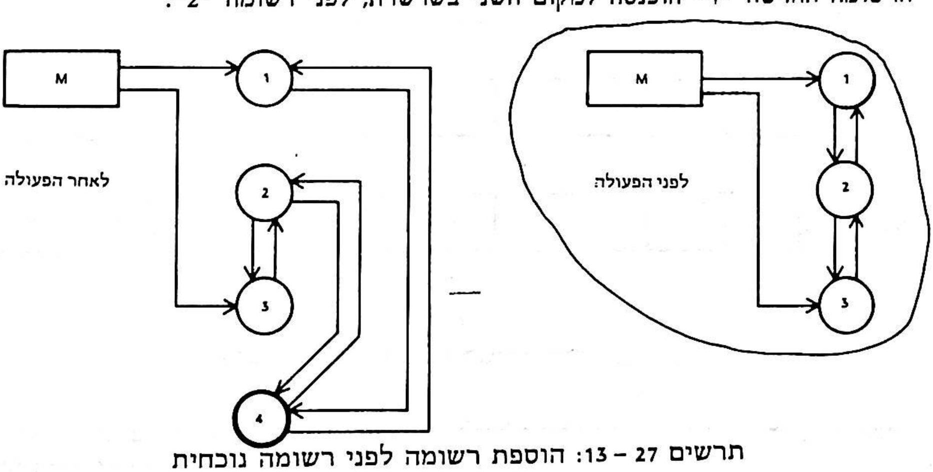
6.3 פקודות לעיבוד קובץ משני

READV: קריאת רשומה בקובץ משני באמצעות מצביע לפנים.
READR: קריאת רשומה בקובץ משני באמצעות מצביע לאחור.
READD: קריאת רשומה לפי המקום היחסי שלה בשרשרת. המקום היחסי נמסר בשדה REFERENCE בפקודת הקריאה.
WRITV: עדכון רשומה בקובץ משני. יש תחילה לקרוא את הרשומה באמצעות אחת הפקודות הקודמות.
ADDVC: הוספת רשומה חדשה בסוף שרשרת. המערכת תעדכן את המצביעים לאחר הפעולה.



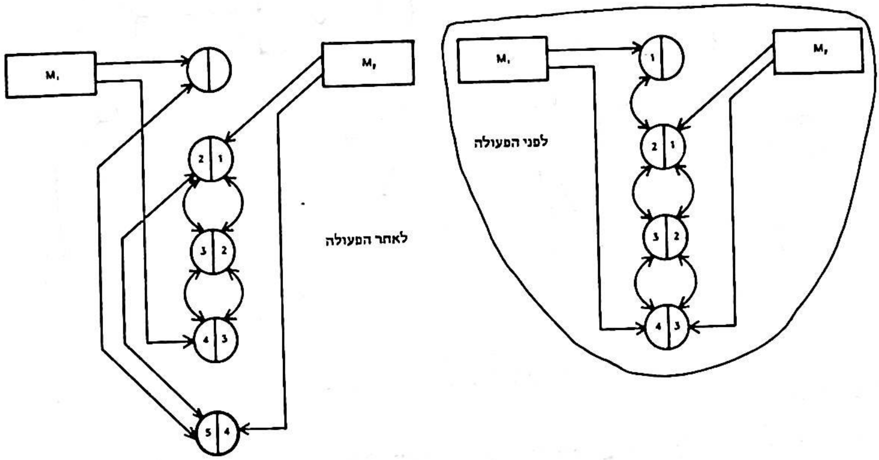
תרשים 26 – 13: הוספת רשומה בסוף שרשרת

ADDVB: הוספת רשומה חדשה לפני הרשומה הנוכחית שהתקבלה בפקודת הקריאה האחרונה. בתרשים 27—13 מוצגת הוספת רשומה לאחר קריאת הרשומה המשנית "2". הרשומה החדשה "4" הוכנסה למקום השני בשרשרת, לפני רשומה "2".



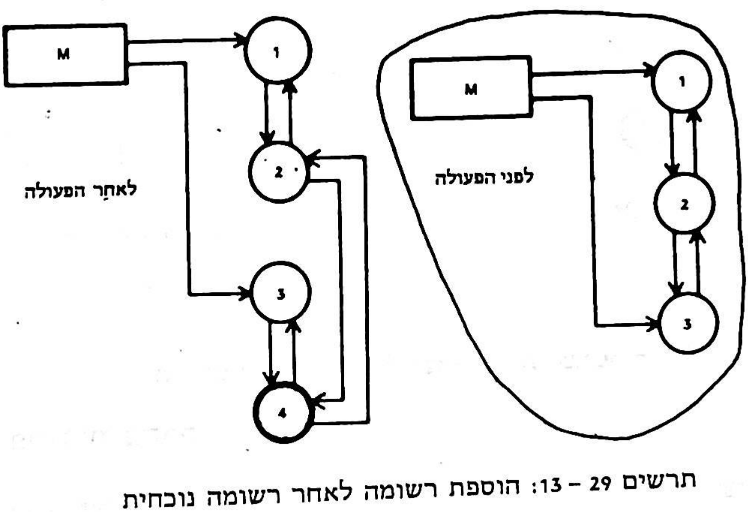
תרשים 27 – 13: הוספת רשומה לפני רשומה נוכחית

תרשים 28—13 מציג דוגמא של קובץ משני אשר קשור לשני קבצים ראשיים. הרשומה החדשה שנוספה "5/4" תהייה קשורה לשתי רשומות ראשיות. בשרשרת הקשורה לקובץ M_1 הרשומה הוכנסה לפי מקומה הלוגי בהתאם לרשומה הנוכחית ועל־כן היא תופשת את המקום השני בשרשרת. אותה רשומה הוכנסה בסוף השרשרת הקשורה לקובץ M_2.



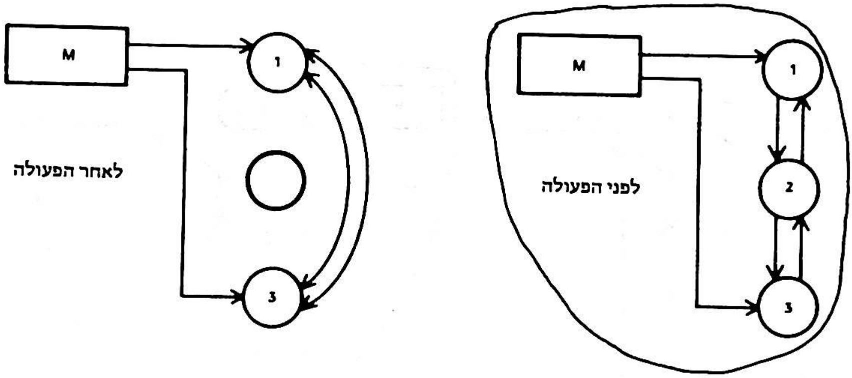
תרשים 28 - 13: הוספת רשומה בשתי שרשרות

ADDVA: הוספת רשומה חדשה לאחר הרשומה הנוכחית שנקבעה בפקודת קריאה אחרונה. בפעולת ההוספה המוצגת בתרשים 29—13 הרשומה הנוכחית היא הרשומה השניה ולכן הרשומה החדשה תוכנס למקום השלישי בשרשרת.



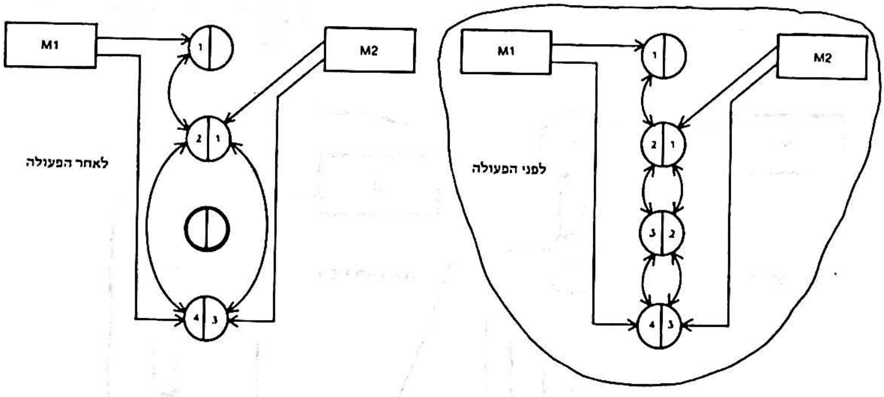
תרשים 29 - 13: הוספת רשומה לאחר רשומה נוכחית

DELVD: ביטול רשומה משרשרת בקובץ המשני. יש לקרוא תחילה את הרשומה באחת מפקודות הקריאה. תרשים 30—13 מציג דוגמא לביטול הרשומה השניה בשרשרת.



תרשים 30 - 13: ביטול רשומה בשרשרת

אם הרשומה המשנית משתתפת במספר קשרים היא מתבטלת בכל הקשרים שהיא משתתפת בהם, כפי שמוצג בתרשים 31—13.



תרשים 31 - 13: ביטול רשומה בשתי שרשרות

6.4 פקודות בקרה

SINON: תוכנית היישום מכריזה באמצעות פקודה זו על עיבוד הקובץ, ומציינת את סוג

העיבוד — קריאה או עדכון. המערכת מפקחת שהתוכנית תבצע רק את הפעולות שמורשות לה. תוכנית יכולה למנוע מתוכניות אחרות לגשת לקובץ שהיא פועלת בו כדי למנוע את עדכונו בו־זמנית על ידי תוכניות שונות. נעילת הקובץ גורמת לכתיבת סימון מיוחד בכותרת הקובץ.

SINOF: שחרור הקובץ על ידי תוכנית היישום לשימוש של תוכניות אחרות.

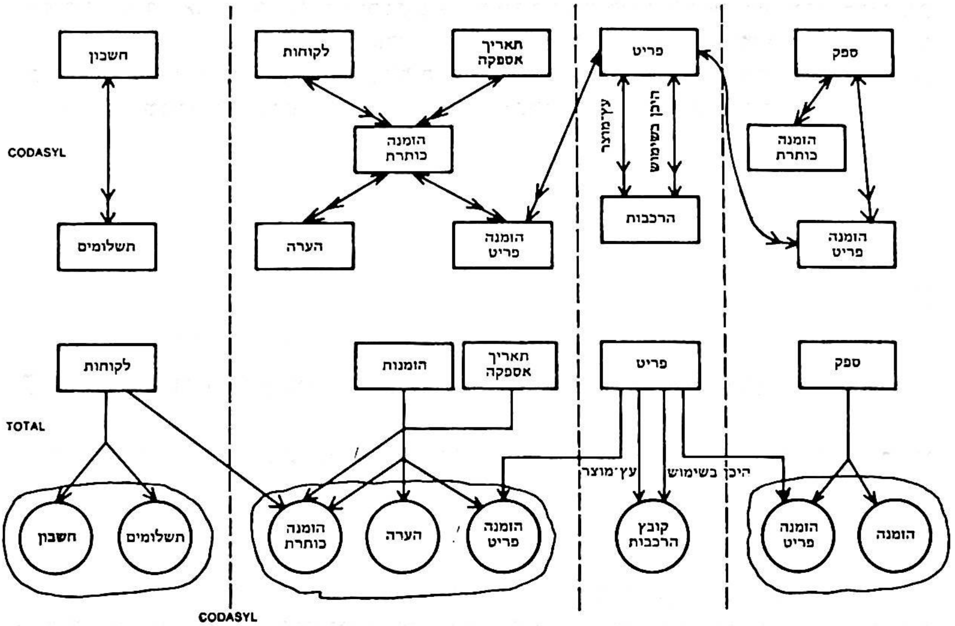
7. השוואה בין מערכת TOTAL לבין המלצות CODASYL

המבנה הבסיסי בהמלצות CODASYL היא **החבורה** (Set). במערכת TOTAL הקשר בין הקובץ הראשי לקובץ המשני מציג מבנה דומה למבנה החבורה, והוא מכיל סוגי קשר שונים: מצביע לפנים, מצביע לאחור ומצביע סמלי לרשומת האב. המשתמש אינו יכול לבחור באילו מצביעים ישתמש בהוראות השונות.

רשומות חדשות מתווספות באופן אוטומטי בסוף השרשרת, אך המשתמש יכול להכניסן במקום שונה אם ציין כך בפקודת העדכון. בהמלצות CODASYL ניתן לקבוע את הסדר של החבורה (LIFO, FIFO, ממויין) בזמן הגדרת הסכימה.

מבנה הרשת ב־TOTAL מורכב יותר מכיון שאין אפשרות לקשרים ישירים בין קבצים ראשיים או משניים. הרשת המתקבלת בהמלצות CODASYL מתאימה יותר למבנה האמיתי של הקשרים בין הנתונים.

תרשים 32—13 מציג דוגמא למאגר נתונים ב־CODASYL וב־TOTAL. מאגר הנתונים ב־TOTAL מכיל קבצים ראשיים וקבצים משניים.



תרשים 32 – 13: מבנה נתונים ב־CODASYL וב־TOTAL

8. סיכום

מערכת TOTAL לניהול נתונים מאפשרת הגדרת קשרים שונים ומגוונים בין הנתונים. היא משתמשת במודל הרשת אך אינה מכילה את כל התכונות המומלצות על־ידי CODASYL. הטיפול בנתונים נעשה בשפה פרוצדורלית והמשתמש חייב להגדיר את המאגרים בצורה מפורשת. המערכת אינה מאפשרת הגדרת תנאי שליפה לרשומות ולכן רוב העיבוד הוא סדרתי לאורך הקשרים. המערכת יכולה לפעול באופן מקוון (On-line) באמצעות מערכת תוכנה לניהול תקשורת כמו CICS, ENVIRON/1 ואחרות. מחולל הדוחות SOCRATES מאפשר עריכת דוחות מבסיס הנתונים.

מערכת TOTAL הינה אחת ממערכות ניהול בסיסי הנתונים הנפוצות ביותר. אפשר להתקינה במחשבים של יצרנים שונים: במערכות יבמ בארכיטקטורת 370, מערכות Honeywel, Univac, NCR, CDC, PDP/11, Burroughs ועוד.

חברת Cincom הכריזה לאחרונה על מערכת TIS לניהול בסיסי נתונים. מערכת זו משתמשת במבנים הפיסיים של מערכת TOTAL אולם משחררת את המשתמשת מטיפול בקבצים ראשיים ומשניים. המשתמש עובד עם רשומות לוגיות (LUV — Logical User View) אשר יכולות להכיל שדות מקבצים ראשיים ומשניים כאחד. שיטה זו מפשטת מאד את הטיפול בבסיס הנתונים מכיון שהמשתמש עובד עם רשומות שטוחות בלבד. מערכת TIS מכילה מילון נתונים פעיל המשמש לאחסון ההגדרות של הקבצים, השדות, הקשרים וה־LUV. מערכת TIS משולבת עם מחולל היישומים MANTIS, מחולל דוחות COMPREHENSIVE RETRIEVAL ושפת שאילתות למשתמש הקצה INTELLIGENT QUERY.

פרק יד

ADABAS —
Adaptable Data Base System

1. מבוא
2. מושגי יסוד
3. מבנה פיסי של בסיס הנתונים
4. מבנים לוגיים
5. שפה להגדרת נתונים
6. שפה לטיפול בנתונים
7. סיכום

1. מבוא

מערכת ADABAS פותחה על ידי חברת Software AG במערב גרמניה ומיועדת בעיקר למחשבי יבמ גדולים. לאחר זמן פותחה מערכת ADABAS-M המיועדת למיני מחשבים ממשפחות PDP ו־VAX של חברת DEC.

ADABAS אינה מניחה מבנה מיוחד כלשהו של הנתונים, כגון מבנה היררכי או רשתי. כל הרשומות במערכת הן רשומות שטוחות רגילות כאשר הגישה אליהן מבוצעת באמצעות **רשימות הפוכות** (Inverted Lists). בסיס הנתונים בנוי מקובץ אחד או יותר אשר מכיל את הרשומות ומקובץ נוסף אשר מכיל את הרשימות ההפוכות. המערכת מאפשרת הגדרת מספר רב של שדות כמפתחות משניים.

אפשר לקשור קבצים זה לזה באמצעות הרשימות ההפוכות על פי תוכן שדה מסויים המופיע ברשימות ההפוכות שלהם. הקישור מאפשר לבנות בסיס הנתונים המתאים למודל הרשתי או המודל ההירככי, למרות תצוגת הנתונים למשתמש אשר דומה לתצוגתם במודל הטבלאי. עם זאת, מערכת ADABAS איננה מערכת טבלאית מכיון שהיא חסרה חלק מהאופרטורים המוגדרים במודל הטבלאי.

2. מושגי יסוד

כדי לתאר את המערכת יש להכיר תחילה מספר מושגי יסוד הנהוגים בה:

— **שדה** (Field): השדה הוא יחידת הנתונים אשר מכיל נתון לוגי כלשהו והוא יכול להתחלק **לתת־שדות** (Sub-Fields). המערכת מבחינה בין שני סוגי שדות: **שדה אלמנטרי** (Elementary Field) **ושדה רב־ערכי** (Multiple-Value Field). שדה אלמנטרי יכול להכיל ערך אחד בלבד בעוד אשר שדה רב־ערכי עשוי להכיל מספר ערכים שונים. ניתן להגדיר **שדה קבוצתי** (Group Field) אשר מורכב ממספר שדות. שדה קבוצתי החוזר על עצמו מספר פעמים בתוך הרשומה הוא **שדה קבוצתי מחזורי** (Periodic Group Field).

— **מפתח** (Descriptor): המפתח הוא ערך שדה, תת־שדה או שדה קבוצתי ברשומה אשר מאפשר גישה אליה באמצעות הרשימות ההפוכות. ניתן להגדיר שדות מפתח אחדים לכל רשומה ובהתאם לכך תהיה לכל אחד מהם רשימה הפוכה. את שדות המפתח ניתן להגדיר בעת יצירת הקובץ או לאחר מכן באמצעות תוכנית שירות מיוחדת. המערכת מאפשרת הגדרת **מפתחות יחודיים** (Unique) אשר אינם מכילים ערכים כפולים. מטרת השימוש במפתחות הינה ליעל את הגישה לרשומות אשר בבסיס הנתונים.

— **רשומה** (Record): הרשומה הינה יחידת התקשורת הבסיסית בין בסיס הנתונים לבין תוכנית היישום. הרשומה מוגדרת כאוסף של מספר שדות בעלי משמעות לוגית מבחינת המשתמש. רשומה אחת יכולה להכיל עד 500 שדות נתונים. כדי לאפשר גמישות בניהול השדות ברשומה מוסיפה המערכת לכל שדה ברשומה את הזיהוי שלו ואת אורכו. בכך היא מאפשרת למשל להוסיף שדות לאזור פנוי ברשומה מבלי שיהיה צורך לטעון מחדש את הקובץ. שיטת ניהול השדות מאפשרת גם לדחוס את הנתונים ולקבל **רשומה דחוסה** (Condensed Record). מטרת הדחיסה הינה בעיקר לחסוך בנפח הקובץ ולהוסיף נתונים על ידי "פינוי" מקום ברשומה מבלי שיהיה צורך לטעון מחדש את הקובץ. בתהליך הדחיסה מסלקת המערכת **מרווחים** (blanks) רצופים בשדות אלפבתיים, מסלקת אפסים מובילים בשדות מספריים והופכת אותם **למבנה ארוז** (Packed). כאשר ישנם **שדות ריקים** (Null Value) רושמת המערכת עובדה זו וחוסכת את המקום שלהם. בדרך כלל יכול תהליך הדחיסה לחסוך עד 50% בשטח אחסון הקובץ. הדחיסה נעשית באופן אוטומטי על ידי המערכת לפני כתיבת הרשומה. אפשר **למנוע** את פעולת הדחיסה כפי שיוסבר בהמשך.

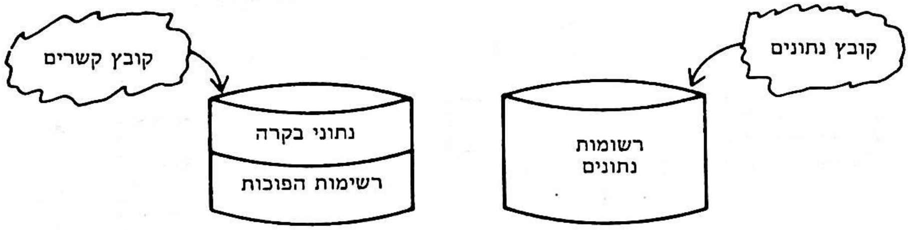
— **קובץ** (File): קובץ הוא אוסף כל המופעים של הרשומות מאותו הסוג. כל קובץ יכול להכיל עד 16 מיליון רשומות שונות בגושים בעלי גודל קבוע. מספר הרשומות לגוש משתנה לפי אורכן ולאחר שהקובץ עובר תהליך דחיסה. בכל גוש משאירים מקום פנוי מסויים לצורך הגדלה עתידית של הרשומות על ידי תוספת שדות, או שינוי בשדות קיימים.

— **בסיס נתונים** (Data Base): בסיס נתונים במערכת ADABAS מכיל עד 255 קבצים שונים אשר כל אחד מהן מכיל סוג רשומה אחד בלבד.

— **מספר סודר פנימי** (ISN—Internal Sequence Number): לכל רשומה פיסית הנרשמת בקובץ נקבע על ידי המערכת מספר סודר פנימי. המספר הסודר נשאר קבוע ולא ניתן לשנותו, אלא רק לבטלו כאשר מבטלים את הרשומה. מטרתו העיקרית לשמש כמצביע אל הכתובת היחסית של הרשומה בקובץ.

3. מבנה פיסי של בסיס הנתונים

מערכת ADABAS מכילה שני סוגי קבצים: **קובץ נתונים** אשר מכיל את רשומות הנתונים (Data Records) **וקובץ הקשרים** אשר מכיל את הרשימות ההפוכות (Inverted Lists) ונתוני בקרה (Control Information).



תרשים 1 – 14: שני סוגי הקבצים ב־ADABAS

3.1 קובץ הנתונים (Data Storage)

קובץ הנתונים הינו קובץ BDAM המכיל את הרשומות באורך משתנה בתוך גושים בעלי גודל קבוע. אחוז מסוים מגודל הגוש נשאר פנוי לטיפול בגידול עתידי של הרשומות בגוש. גודל השטח הפנוי נקבע בזמן בניית הקובץ.

3.2 קובץ הקשרים (Associator)

קובץ הקשרים מכיל את נתוני הבקרה, טבלאות הניהול ואת הרשימות ההפוכות. נפחו כרבע מנפח האחסון של קובץ הנתונים. המרכיבים העיקריים בקובץ הקשרים הם:

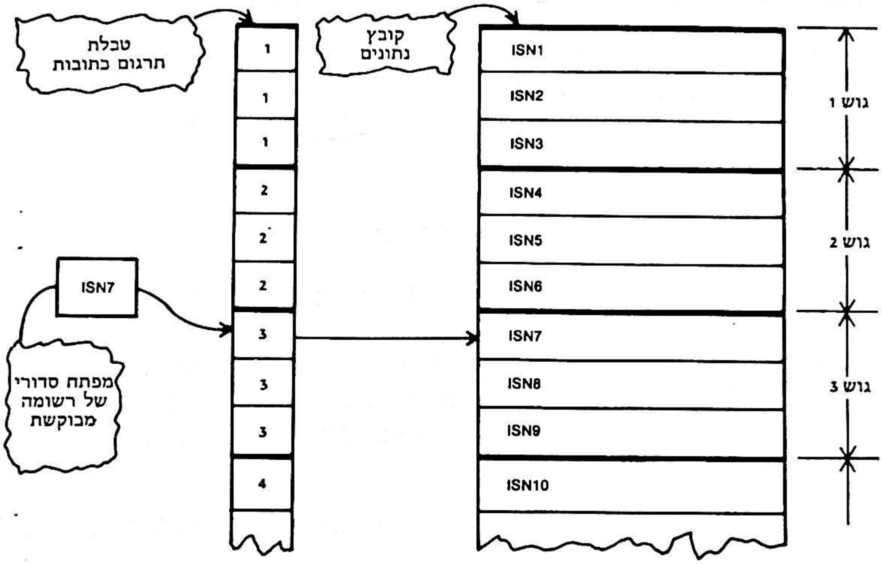
— **טבלאות לתיאור רשומות** (Field Description Tabels): הטבלאות מכילות את ההגדרה הלוגית של הרשומות: לכל שדה ברשומה ניתן התיאור, התכונות והסיווג. המשתמש יכול לפנות לטבלה זו באמצעות פקודות מיוחדות.

— **טבלאות לניהול שטחי אחסון** (Storage Management Tabels): הטבלאות מיועדות לסייע בניהול שטח האחסון של בסיס הנתונים ולמטרה זו הן מכילות מידע על תכולת שטחי האחסון של הקבצים השונים. כאשר נדרש לקובץ שטח אחסון מעבר להקצאה הקיימת, השגרות לניהול שטחי האחסון מאתרות ומקצות אותו מתוך שטח האחסון הרזרבי שבפיקוחן. כאשר מבטלים רשומה או קובץ מתבצע רישום מתאים של השטח שהתפנה לשם שימוש חוזר.

— **מתרגם כתובות** (Adddress Convertor): בזמן טעינת הרשומות לקובץ מוסיפה המערכת לכל רשומה את המספר הסדורי הפנימי (ISN) שלה. מספר הגוש הפיסי שבו נרשמת הרשומה נרשם בטבלה מיוחדת הנקראת **טבלה לתרגום כתובות**. טבלה זו הופכת למעשה את ה־ISN למפתח לוגי ולא פיסי מאחר והוא איננו מכיל את הכתובת הפיסית של הרשומה אלא את מספרה

הסדורי בתוך הקובץ. כאשר משנים מקום של רשומה לאחר שהיא גדלה ואין לה מקום בגוש המקורי שלה, אין צורך לשנות את המספר הסודר שלה. הרשומה עוברת עם ISN שלה ובטבלה מציינים את מספר הגוש החדש שבו היא נמצאת. שיטה זו קרויה שיטת **מיעון עקיף** (Indirect Addressing) לרשומות.

טבלת תרגום הכתובות מכילה כניסה אחת לכל רשומה בקובץ הנתונים והמפתח הסדורי הפנימי של הרשומה משמש לה כאינדקס הגישה. כלומר, במקום החמישי בטבלה נמצא את הכתובת הפיסית של הרשומה שה־ISN שלה הוא חמש וכדומה. הכתובת הפיסית הינה כתובת הגוש הפיסי שבו מאוחסנת הרשומה.

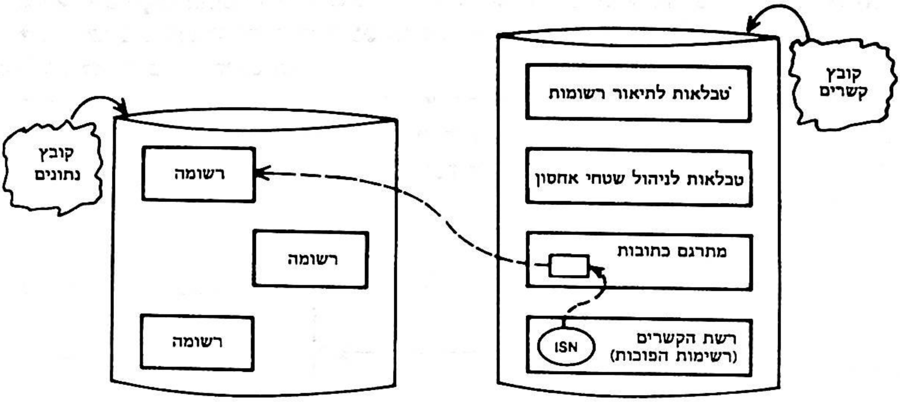


תרשים 2 – 14: חיפוש רשומה באמצעות טבלה לתרגום כתובות

— **רשת הקשרים** (Association Network): רשת הקשרים כוללת את הרשימות ההפוכות הדרושות לאיתור רשומה לפי מפתח. לכל מפתח מנוהלת רשימה הפוכה נפרדת. הרשימה מכילה עבור כל ערך של המפתח את רשימת המספרים הסודרים (ISN) של הרשומות שבהן מופיע ערך זה ומונה את מספר הרשומות האלה. הרשימה ממויינת לפי ערך המפתח.

בתרשים 4—14 מוצג לדוגמא קובץ ספקים אשר מכיל רשומה אחת עבור כל ספק ואת הרשימות ההפוכות שלו. הרשומה בקובץ כוללת שדות אחדים: מספר ספק, שם ספק, כתובת, סוג ספק ועוד. לקובץ הספקים הוגדרו שלשה שדות מפתח: מספר ספק, שם ספק וסוג ספק. המערכת תבנה שלש רשימות הפוכות, אחת עבור כל מפתח של הרשומה. הרשימה תכיל את ערך המפתח ואת רשימת המספרים הסודרים (ISN) של הרשומות המכילות ערך זה. הרשימה ההפוכה מיועדת לאיתור רשומות לפי תוכן שדה המפתח, ללא צורך בגישה אל הרשומות עצמן. כדי להקל על החיפוש בונה המערכת לכל אחת מהרשימות ההפוכות רמות אינדקס אחדות (עד 6) בהתאם

למספר המפתחות, מספר הרשומות בקובץ ואורך שדה המפתח. האינדקסים "שקופים" למשתמש ומנוהלים אוטומטית על ידי המערכת.



תרשים 3 – 14: מבנה קובץ הקשרים

קובץ נתונים

ISN	מספר ספק	שם ספק	כתובת	סוג ספק
ISN1	105	תדיראן	גבעת שמואל	1
ISN2	124	טלרד	אזור תעשיה לוד	1
ISN3	185	ביטקום	קרית עתידים, תל־אביב	2
ISN4	207	דגם מערכות	קרית עתידים, תל־אביב	1
ISN5	310	אילתא	אשרוד	2

מספר ספק	מונה	רשימת ISN
105	1	ISN1
124	1	ISN2
185	1	ISN3
207	1	ISN4
310	1	ISN5

שם ספק	מונה	רשימת ISN
אלתא	1	ISN5
ביטקום	1	ISN3
דגם מערכות	1	ISN4
טלרד	1	ISN2
תדיראן	1	ISN1

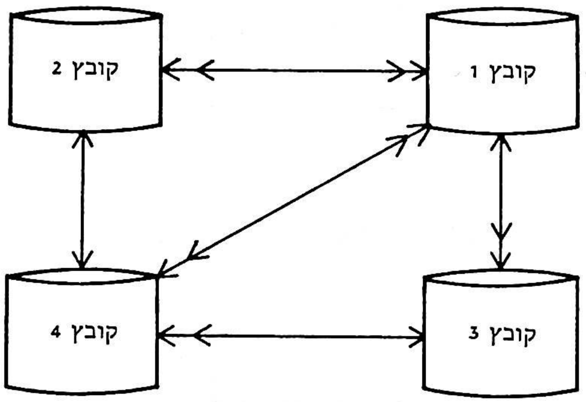
סוג ספק	מונה	רשימת ISN
1	3	ISN4,ISN2,ISN1
2	2	ISN5,ISN3

תרשים 4 – 14: קובץ ורשימות הפוכות

4. מבנים לוגיים

לאחר הגדרת הרשומות הבודדות המרכיבות את הקבצים בבסיס הנתונים ניתן לבצע את **זיווג הקבצים** (File Coupling) לשם בניית רשתות או היררכיות. קישור בין רשומות של שני קבצים כלשהם מתאפשר על פי שדה משותף אשר מופיע בשתיהן. ניתן לבצע קישור או לבטלו בכל עת לאחר בניית הקבצים.

הקשרים בין הקבצים מיוצגים על ידי טבלאות ממויינות של מספרים סודרים אשר נשמרות בקובץ הקשרים. אפשר לקשור קובץ אחד עד ל־80 קבצים אחרים, וכל רשומה יכולה להיות קשורה למספר בלתי מוגבל של רשומות בקובץ אחר.



תרשים 5 – 14: קישור קבצים

בתרשים 6—14 מוצגת דוגמא לקישור בין שני קבצים, קובץ ספקים וקובץ פריטים. נניח שקיים ביניהם קשר רב—רב־ערכי, כלומר כל ספק יכול לספק מספר פריטים וכל פריט יכול להיות מסופק על ידי מספר ספקים. לשם כך יהיה ברשומת הספק שדה רב־מופעי אשר מכיל את מספרי הפריט המסופקים על ידו, וברשומת הפריט יהיה שדה רב־מופעי המכיל את מספרי הספק שלו. את קישור הקבצים אפשר לבצע על פי השדות "מספר ספק" ו"מספר פריט" אשר מופיעים ברשומות שני הקבצים. הקשר שהתקבל הוא רב—רב־ערכי דו־כיווני.

קישור שני קבצים זה לזה מאפשר הגדרת מבני נתונים מורכבים ביותר לשם הצגת מודל משתמש (User View) רשתי או היררכי בהתאם לצורך.

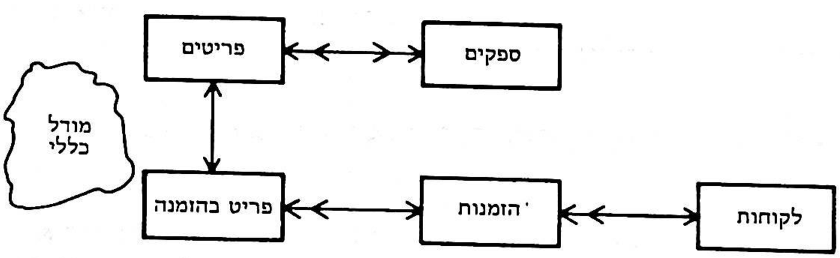
מספר סורר	מספר ספק	שם ספק	פריטים לספק			
ISN1	12	תדיראן	A12	B17	B98	D65
ISN2	13	אלתא	A15	B17	C11	D65
ISN3	15	טלרד	B17	C15	D65	
ISN4	21	ביטקום	C11	C15	D65	

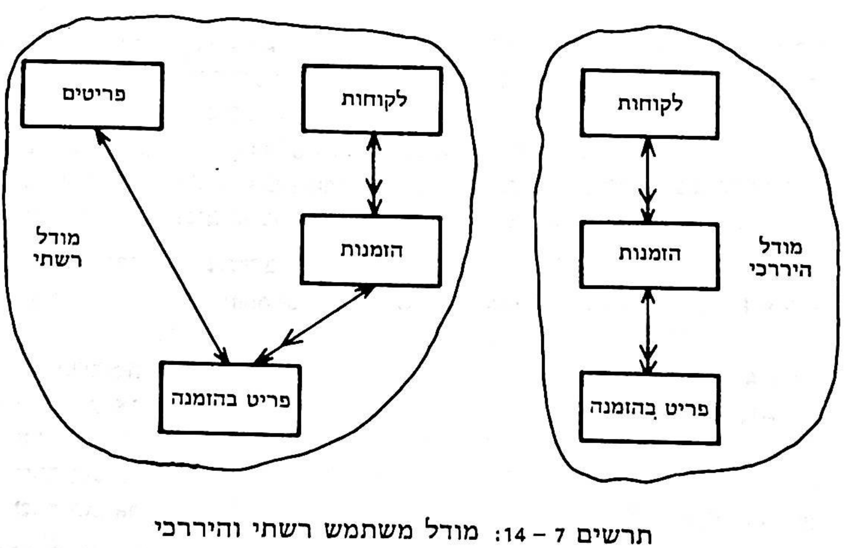
קובץ ספקים

מספר סורר	מספר פריט	תיאור פריט	ספקים לפריט					
ISN1	A12	בורג	12	13				
ISN2	A15	בורג	13					
ISN3	B17	אום	12	13	15			
ISN4	B91	מפסק	21					
ISN5	B98	מפסק	12					
ISN6	C11	בורג	13	21				
ISN7	C15	מפסק	15	21				
ISN8	D65	נורית	12	13	15	21		

קובץ פריטים

תרשים 6 – 14: קשר רב – רב־ערכי בין קובץ ספקים לבין קובץ פריטים





תרשים 7 – 14: מודל משתמש רשתי והירככי

5. שפה להגדרת נתונים

הגדרת הנתונים הינה תולדה של הגדרת הקבצים, הרשומות, השדות והקשרים שבין הקבצים. לכל קובץ מספר מזהה אשר נקבע על־ידי המערכת בזמן טעינתו הראשונית.

השדה הינו היחידה הבסיסית החשובה ביותר מכיון שהוא מכיל את הנתונים. הוא יכול להיות שדה מפתח שבאמצעותו יוצרים את הקשר בין רשומות וקבצים. המשתמש צריך להגדיר כל שדה אך אין הוא נזקק תמיד לכל הפרמטרים מכיון שאינם דרושים או שקיימות עבורם ברירות מחדל קבועות במערכת. הפרמטרים "רמה", "שם", אורך ו"סוג" חייבים להופיע תמיד. הגדרת השדה כוללת את הפרמטרים הבאים:

— **רמת השדה** (Field-Level): מספר בן שתי ספרות אשר מגדיר את רמת השדה ברשומה, כמקובל בשפות תכנות גבוהות כגון קובול. קיימת אפשרות של הקבצת שדות לשם יצירת מבנה הירכי.

— **שם השדה** (Field-Name): לכל שדה ניתן שם סמלי באורך של שני תוים בלבד.

— **אורך השדה** (Field-Length): אורך השדה המקורי לפני הדחיסה. המערכת מבצעת דחיסה של כל הרשומות לפני כתיבתן בקובץ אלא אם **נדרש** לא לבצע זאת.

— **סוג השדה** (Field-Type): המערכת מכירה מספר סוגי שדות: מספרי ארוז (Packed), מספרי לא־ארוז (Unpacked), אלפנומרי, בינרי, נקודה־צפה.

— **אופציות נוספות**: במסגרת אופציות אלה ניתן להגדיר תכונות נוספות של השדה, להלן כמה מהאפשרויות:

א. **שדה מפתח** (Descriptor): השדה מוגדר כשדה מפתח אל הרשומה לשם חיפוש, מיון ועיבוד לוגי סדרתי.

ב. **שדה רב־ערכי** (Multiple-Value): השדה מכיל מספר ערכים שונים. לדוגמא "שם רחוב" הוא שדה אחד אשר עשוי להכיל מספר מילים אשר מייצגות ערכים נפרדים כמו למשל ברחוב ששמו "חיים נחמן ביאליק". לצורך שליפה אפשר להגדיר חלק מהערכים של השדה, כמו למשל "נחמן" או "ביאליק". דוגמא נוספת לשדה רב־ערכי היא שדה "מילות מפתח" של מסמך.

ג. **שדה מחזורי** (Periodic Field): השדה עשוי להופיע ברשומה מספר משתנה של פעמים, כמו למשל בוקטור חד־ממדי. הפנייה לשדה מחזורי תיעשה על ידי ציון האינדקס הקובע את מקומו היחסי בוקטור.

ד. **שדה באורך קבוע** (Fixed Length): אופציה זו **מבטלת** את דחיסת השדה.

ה. **שדות ריקים** (Null-Value Distinction): המערכת עורכת דחיסת נתונים ולכן יש לציין אם שדה שבו אפסים או מרווחים הוא **ריק** (Null) או שאלו הם נתונים.

מבנה כללי להגדרת רשומה ב־ADABAS:

דוגמאות:	Field-Value, Field-Name, Field-Length, Field-Type, Options...
— שם הרשומה AA (אין פרוט)	01, AA
— שדה A1, אורך 10, מבנה ארוז (P), מפתח	02, A1, 10, P, DE
— שדה A2, אורך 5, ארוז	02, A2, 5, P
— שדה A3, אורך 10, אלפנומרי	02, A3, 10, A

בדוגמא זו רמת הרשומה היא 01 ורמת כל השדות היא 02.

6. שפה לטיפול בנתונים

השפה לטיפול בנתונים נותנת למשתמש שירותים של עיבוד ואחזור נתונים.

6.1 עיבוד נתונים

המתכנת יכול לקבל את שירותי מערכת התוכנה ADABAS באמצעות הוראות CALL אשר כתובות בשפה העילית המארחת COBOL או PL/1 מבנה ההוראה:

CALL 'ADABAS' USING CONTROL—BLOCK, FORMAT—BUFF, RECORD—BUFF, SEARCH—BUFF, VALUE—BUFF, UPDATE—BUFF.

כל הפרמטרים שלאחר USING הם שמות משתנים. אין צורך להשתמש בכל הפרמטרים.

— **שטח בקרה** (Control-Block): בשטח הבקרה נמסרת הפקודה שיש לבצע, זיהוי המשתמש, מספר הקובץ, גודל המאגרים לקלט/פלט ואופציות נוספות כגון מיון. בשטח זה נמסר שם שדה שמוגדר בתוכנית כדי לקבל בו את **קוד־החזרה** (Return-Code) לאחר ביצוע הפעולה על ידי ADABAS. הפקודות האפשריות הן:

א. S1: הפקודה מחזירה את רשימת כל המספרים הסודרים (ISN) של הרשומות אשר עונות לקריטריון החיפוש שנמסר ב־Search-Buff. הרשומה מועברת לשטח המצויין בפקודה ואם אין בו מקום מספיק, ירשמו המספרים הגולשים בשטח זמני ואחר־כך יועברו לשטח העבודה. מספר המופעים של רשומות שמתאימות לקריטריון החיפוש נמבר בשדה בשטח הבקרה. מספר זה מאפשר לקבוע את מספר החזרות הדרוש על פקודת S1 כדי לקרוא את רשימת כל המספרים הסודרים.

ב. S2: פקודה זהה לפקודה S1 אך היא מחזירה רשימה ממויינת של המספרים הסודרים לפי תוכן שדות המפתח.

ג. L1: פקודה זו מעבירה רשומה מהקובץ אל שטח הרשומה בתוכנית (Record-Buff). משתמשים בפקודה זו לאחר פקודת חיפוש אם ISN של הרשומה לא ידוע. אם ה־ISN של הרשומה ידוע, יש תחילה להעביר את תוכנו לשדה בשטח הבקרה.

ד. L2: הוראה לקריאת הרשומות לפי הסדר הפיסי שלהן בקובץ.

ה. L3: הוראה לקריאת הרשומות לפי סדר של אחד מהמפתחות שלהן.

ו. L9: שליפת כל הערכים של מפתח מסויים ומספר המופעים. במקרה זה מתבצעת גישה רק לקובץ הקישור. ניתן להשתמש בפקודה זו על מנת לבדוק כמה רשומות קיימות בקובץ עבור ערך מסויים של המפתח.

ז. N1: פקודה זו מוסיפה רשומה חדשה לקובץ. הערכים של שדות הרשומה צריכים להיות ב־Update-Buff, והמבנה של הרשומה צריך להיות ב־Format-Buff.

ח. A1: פקודה לעדכון שדה אחד או יותר ברשומה קיימת. בדרך כלל ניתנת פקודה זו לאחר פקודת חיפוש. יש למסור בשטח הבקרה את שמות השדות שיש לעדכן ואת הערכים החדשים.

ט. A2: פקודה זו מאפשרת להוסיף שדות חדשים לרשומה קיימת, כלומר לעשות שינוי של הסכימה. הפקודה "מגדילה" את הרשומה אך אינה רושמת בה ערכים חדשים כלשהם. הערכים ירשמו בפקודת A1.

י. E1: פקודה לביטול רשומה מהקובץ. את ה־ISN של הרשומה מאתרים בדרך כלל

באמצעות פקודת חיפוש.

— **מאגר מבנה הרשומה** (Format-Buffer): המאגר מכיל את מבנה הרשומה או את שמות השדות שיש לשלוף. המאגר מכיל רצף של שמות השדות ואורכם.

— **מאגר הרשומה** (Record-Buffer): מכיל את הרשומה שנשלפה מהקובץ בהתאם למבנה שנקבע במאגר מבנה הרשומה.

— **מאגר חיפוש** (Search-Buffer): מכיל את קריטריון החיפוש, אשר מורכב ממספר הקובץ, שם שדה המפתח ותנאי בוליאני כלשהו.

— **מאגר ערכים** (Value-Buffer): מכיל את תוכן שדות המפתח שהוגדרו במאגר החיפוש.

— **מאגר עדכון** (Update-Baffer): מכיל את תוכן השדות עבור בניית רשומה חדשה או שדות שמעדכנים רשומה קיימת.

6.2 שאילתות

שפת השאילתות ADASCRIPT מונעת את הצורך להשתמש בשפה מארחת ובהוראות המערכת המורכבות כפי שראינו קודם. שפת השאילתות דומה במבנה החיצוני לשפת קובול או לשפה אנגלית והיא מאפשרת לבצע פעולות פשוטות של שליפה, עדכון והצגת שדות מתוך הרשומה בשיטה אינטראקטיבית.

לדוגמא, הוראת FIND לאיתור רשומות:

```
FIND IN SUPPLIER—FILE ALL RECORDS
   WITH SUPPLIER—ID = 305
```

שליפת רשומה על פי תנאי בוליאני:

```
FIND IN SUPPLIER—FILE ALL RECORDS
   WITH SUPPLIER—TYPE > 1
```

את הרשומות הנשלפות ניתן למיין לפי שדה כלשהו ברשומה:

```
FIND IN SUPPLIER—FILE ALL RECORDS
   WITH SUPPLIER—TYPE > AND SORT BY SUPPLIER—NAME
```

אם קובץ הספקים קשור לקובץ הפריטים (File Coupling) ניתן להציג שאילתות מורכבות המבוססות על קשר זה. לדוגמא:

— שליפת כל הפריטים המסופקים ע"י ספק נתון ואשר מחירם גבוה מ־10000 שקל. את הפריטים יש להציג בצורה ממויינת לפי שם הפריט:

```
FIND ITEMS WITH PRICE > 10000 AND
   COUPLED TO SUPPLIER—FILE
   WITH SUPPLIER—ID = 305
   SORT BY ITEM—DESCRIPTION
   DISPLAY ITEM—DESCIRITON, ITEM—ID, SUPPLIER—NAME.
```

— ניתן לשאול בצורה דומה שאילתא סימטרית המבקשת את רשימת כל הספקים המספקים פריט מסויים:

```
FIND SUPPLIER—FILE AND
   COUPLED TO ITEMS
   WITH ITEM—ID = 'ABC'
   SORT BY SUPPLIER—NAME
   DISPLAY SAPPLIER—NAME, SUPPLIER—ID.
```

ניתן להציג שאילתות עם תנאי שליפה המתייחס לחמישה קבצים שונים הקשורים ביניהם.

השימוש ב־ADASCRIPT נוח מאד בגלל צורת הצגת השאילתא ובגלל מודל הנתונים הפשוט שבו משתמשת מערכת ADABAS לעומת הצורה הפרוצדורלית של השפה לטיפול בנתונים.

7. סיכום

ADABAS היא מערכת ניהול בסיסי נתונים אשר משתמשת במודל נתונים פשוט מאד וברשימות הפוכות. אפשר להוסיף מסלולי גישה בכל נקודת זמן. כל הקשרים בין הנתונים מיושמים בקובץ נפרד ואין להם כל ביטוי בקובץ הנתונים עצמו. המערכת מבצעת דחיסה אוטומטית של הרשומות כדי לחסוך בשטח אחסון. אפשר להשתמש בסיסמאות ומפתחות נוספים לשם בטיחות ואפשר לבצע הצפנה אוטומטית של הנתונים.

בנוסף למערכת הבסיסית של ניהול בסיסי נתונים כוללת המערכת תת־מערכות נוספות לשם שירותים בסביבת עיבוד בתקשורת ובהידברות:

— ADASCRIPT שפת שאילתות דמויית אנגלית לשם אחזור נתונים וביצוע עיבודים פשוטים,

— ADACOM מחולל דוחות,

— NATURAL — מעבד שפה הידברותית המכיל מספר תכונות של ADASCRIPT ו־ADACOM. NATURAL משמש למטרות הבאות: תכנות שאילתות מזדמנות, מחולל דוחות, הגדרת מסכים ופיתוח יישומים מקוונים.

— ADAMINT שפת־מקרו המספקת אפשרויות פיתוח שגרות לגישה לבסיס הנתונים ומגדילה בכך את אפשרויות הבקרה והבטיחות של בסיס הנתונים.

פרק טו

SQL/DS —
Structured Query Language/Data System

1. מבוא
2. הגדרת נתונים
3. טיפול בנתונים
4. בטיחות נתונים
5. סיכום

1. מבוא

מאז תחילת שנות ה־70 הושקעו מאמצים רבים בפתוח מערכות לניהול בסיסי נתונים המבוססות על המודל הטבלאי. אחת המערכות הנסיוניות המפורסמות היא מערכת התוכנה System R שפותחה במעבדות המחקר של חברת יבמ. System R הינה יישום של המודל הטבלאי שנוסח על ידי Dr E.F. Codd. אחת התרומות החשובות של המחקר היא פיתוח שפה לא־פרוצדורלית — SQL (Structured Query Language) אשר מיועדת לשתי מטרות, הן להגדרת נתונים והן לטיפול בהם. השפה שפותחה פורסמה בספרות המקצועית ואומצה גם על ידי חברת Relational Software במערכת ORACLE אשר פותחה על ידה.

אחת המטרות העיקריות בפיתוח SQL היתה פשטות הטיפול בנתונים. ואמנם, אחת התכונות המעניינות ביותר בשפה זו היא הניווט האוטומטי לשם אחזור של הנתונים. ניווט זה מאפשר למשתמש **להגדיר מה הוא רוצה לקבל** מתוך בסיס הנתונים **ולא איך לקבל** את הנתונים. רוב השפות האחרות מחייבות את המשתמש לקבוע "איך" להגיע אל הנתונים באמצעות אוסף של אופרטורים המופעלים בעיתוי מתאים ("קרא רשומת שורש", "התקדם לרשומה הבאה" וכדומה). המשתמש ב־SQL אינו צריך לדעת כיצד הנתונים מאוחסנים בעוד המשתמש במערכות רשתיות או הירככיות חייב להכיר את שיטת אחסון הנתונים. כתוצאה מכך יכולים משתמשים שאינם אנשי מקצוע בענ"א ללמוד את המערכת בקלות ובפשטות יחסית לשם ניצול משאביה באופן עצמאי, ותוכניתני יישומים מנוסים יכולים להגדיל את יעילותם על ידי ניצול מרחב התמרון והאפשרויות.

חברת יבמ פיתחה שתי מערכות לניהול בסיסי נתונים אשר מבוססות על המודל הטבלאי: SQL/DS למערכות DOS/VSE ומערכות VM/CMS, ו־DB2 למערכות MVS. DB2 הינה מערכת מקיפה ומתקדמת יותר אך עקרונות ניהול בסיס הנתונים והשימוש בשפה הינם כמו ב־SQL/DS. בהמשך הפרק נסקור את צורת הגדרת הנתונים והטיפול בהם אשר מתאימים לשתי המערכות.

2. הגדרת נתונים

שפת SQL מאפשרת את **הגדרת הטבלאות** המרכיבות את בסיס הנתונים בצורה פשוטה מאד. פשטות זו נובעת מהעובדה שמבנה הנתונים אינו מכיל מצביעים, אינדקסים או מסלולי גישה כלשהם מוגדרים.

הגדרת טבלאות

בסיס הנתונים במודל הטבלאי מורכב מטבלאות אשר יש להגדיר בעת בניית בסיס הנתונים או לאחר מכן, בכל עת שרוצים להוסיף טבלה חדשה. הגדרת טבלה כוללת את שמה, את שמות השדות (עמודות) המרכיבים אותה ואת התכונות של השדות. בתרשים 1—15 מוצג מבנה של שלש טבלאות שעל פיהן נוכל לרשום את הסכימה הטבלאית של בסיס הנתונים:

ספקים (מספר־ספק, שם־ספק, כתובת)
פריטים (מספר־פריט, תיאור־פריט)
הזמנות (מספר־ספק, מספר־פריט, מחיר, כמות)

ספקים SUPPLIER

S-ADDRESS	S-NAME	S-NUMBER
כתובת	שם־ספק	מספר־ספק

פריט ITEM

ITEM-DESCRIPTION	ITEM-NUMBER
תיאור פריט	מספר פריט

הזמנות ORDERS

QUANTITY	PRICE	IT-NUMBER	SUPP-NUMBER
כמות	מחיר	מספר פריט	מספר ספק

תרשים 1 - 15: הגדרת טבלאות

הגדרת טבלה נעשית בפקודה CREATE TABLE. השמות של השדות בטבלאות השונות יכולים להיות שונים או שהם יכולים להיות שוים. במקרה האחרון יהיה צורך לרשום את שם הטבלה כמציין לשם השדה כאשר מטפלים בנתונים, למשל ITEM.NUMBER או ORDERS.NUMBER כאשר "מספר־פריט" הוא NUMBER בשתי טבלאות שונות. בהוראות להגדרת הטבלאות שבתרשים 1—15 השתמשנו בשמות שדות שונים.

```
CREATE TABLE SUPPLIER  (S—NUMBER          INTEGER NOT NULL,
                        S—NAME            CHAR (30),
                        S—ADDRESS         CHAR (40))
CREATE TABLE ITEM      (ITEM-NUMBER       INTEGER NOT NULL,
                        ITEM—DESCRIPTION  CHAR (45))
CREATE TABLE ORDERS    (SUPP—NUMBER       INTEGER NOT NULL,
                        IT—NUMBER         INTEGER NOT NULL,
                        PRICE             DECIMAL (6.2),
                        QUANTITY          DECIMAL (8.2) )
```

בכל שורה בכל טבלה חייב שדה אחד לפחות להיות שדה־זיהוי או שדה־מפתח ולכן הוא חייב להכיל ערך כלשהו ״לא־אפס״ (NOT NULL). משתנים מספריים אשר מוגדרים INTEGER תופסים מילה שלימה (Fullword). אפשר להגדיר משתנה שלם SMALLINT התופס חצי מילה (Halfword).

מערכת SQL מאפשרת הוספת עמודות חדשות לטבלאות קיימות באמצעות פקודת ALTER TABLE. הרחבת הטבלאות נעשית באופן דינמי, כלומר במהלך העבודה, ואפשר להשתמש מיד בטבלה המורחבת. השדות בעמודות שנוספו הם ריקים (Null) ורישום הנתונים בהם נעשה בפקודות עדכון רגילות של המערכת.

לדוגמא, אם יש להוסיף לטבלת הפריטים עמודה חדשה אשר מכילה את סוג הפריט (תו אחד אלפנומרי) נרשום:

```
ALTER TABLE ITEM
ADD ITEM-TYPE CHAR (1)
```

3. טיפול בנתונים

מערכת SQL מאפשרת גישה לבסיס הנתונים ממסוף באופן אינטראקטיבי, מתוכנית מקוונת או מתוכנית אצווה. הדוגמאות שבהמשך אינן עוסקות במבנה ההוראות להצגת הנתונים ולא בצורת השימוש בהן. לפיכך, הצגת פלט פירושו הצגה במסוף או במאגר התוכנית.

לשליפת נתונים בשפת SQL משתמשים בהוראה SELECT בעלת המבנה הכללי הבא:

SELECT	attribute list	שמות שדות שיש לשלוף (או ★)
FROM	table names	שם טבלה, או טבלאות, שמתוכן יש לשלוף
[WHERE	qualification]	תנאי שליפה
[ORDER BY	attribute-name]	סדר מיון

[ ] מסמן משפט אופציונלי

בנוסף לעוצמה הרבה של פקודת SELECT היא מאופיינת בכך שהיא שולפת מספר שורות (רשומות) בבת־אחת ולא רשומה בודדת כמו בשפות האחרות. כלומר, היא פועלת על **חבורות** (Sets) ומפיקה כפלט חבורה, שהיא למעשה רשימה או דוח שלם.

נציג את שפת SQL באמצעות מספר דוגמאות מבלי לנסות להגדיר אותה בצורה פורמלית. הגדרה מפורטת ניתנה בפרק אשר דן במודל הנתונים הטבלאי. כל הדוגמאות מבוססות על בסיס נתונים טבלאי המכיל את שלש הטבלאות הבאות אשר מוצגות גם בתרשים 1—15:

ספקים (<u>מספר ספק</u>, שם ספק, כתובת)
פריטים (<u>מספר פריט</u>, תיאור פריט)
הזמנות (<u>מספר ספק, מספר פריט</u>, מחיר, כמות)

3.1 שאילתות

שאילתא הינה דרישה לאחזור נתונים מבסיס הנתונים לפי מאפיינים כלשהם. המשתמש במסוף או המתכנת יכול להציג שאילתות שונות, פשוטות ומורכבות יותר אשר תשלופנה נתונים מטבלה אחת או יותר. להלן מספר דוגמאות:

— יש לקבל את רשימת **כל** הפריטים כשהם ממויינים לפי שם פריט.

מציין שיש להציג את כל השדות בטבלה

```
SELECT    *
FROM      ITEMS
ORDER BY  ITEM-DESCRIPTION
```

שדה המיון

— יש לקבל את רשימת כל ההזמנות כשהן ממויינות לפי מספר פריט ובתוכו לפי מספר־ספק. מתוך הטבלה יש להציג רק את מספר־הפריט, מספר־הספק ומחיר־הפריט־בהזמנה.

```
SELECT    IT-NUMBER, SUPP-NUMBER, PRICE
FROM      ORDERS
ORDER BY IT-NUMBER, SUPP-NUMBER
```

— יש לקבל את רשימת כל ההזמנות של ספק 109 אשר בהן מחירי הפריטים בין 1000 ל־2000 שקל.

```
SELECT    *
FROM      ORDERS
WHERE     SUPP-NUMBER = 109 AND
          PRICE BETWEEN 1000 AND 2000
ORDER BY IT-NUMBER
```

כתנאי שליפה אפשר לרשום ביטוי חשבוני במקום שם של שדה או במקום ערך קבוע כלשהו. לדוגמא:

WHERE PRICE > COST*1.2

— יש לדעת תיאור הפריט, מחיר הפריט וכמות בהזמנה של כל הפריטים המסופקים על ידי ספק 110 שבהן המחיר נע בין 1000 ל־2000 שקל. שאילתא זו שונה משאילתות קודמות בכך שאת השדה "תיאור ־הפריט" צריך לשלוף מטבלת הפריטים. הקשר בין שתי הטבלאות מתבצע על ידי השוואת שני השדות של "מספר־פריט" בשתי הטבלאות. קריטריון הקשר נרשם ברשימת תנאי השליפה WHERE.

```
SELECT    ITEM-DESCRIPTION, ITEM-NUMBER, PRICE, QUANTITY
FROM      ORDERS, ITEM
WHERE     SUPP-NUMBER = 110 AND
          PRICE BETWEEN 1000 AND 2000 AND
          IT-NUMBER = ITEM-NUMBER
ORDER BY ITEM-DESCRIPTION
```

שאילתא זו מדגימה את עוצמתה הרבה של SQL. ניסוח שאילתא כזו בשפה פרוצדורלית רגילה הינו מורכב למדי ומחייב מיומנות תכנותית רבה והכרת מבנה הנתונים. לשם כך פותחו במערכות אלו שפות שאילתות אשר "מסתירות" מהמשתמש את מורכבות המערכת ומאפשרות לו גישה פשוטה יותר.

— יש לקבל את שם הספק וכתובתו, תיאור הפריט והמחיר עבור כל ההזמנות של פריט A45B1 שמחירו גבוה מ־2500 שקל. זוהי דוגמא לשליפת נתונים מתוך שלש טבלאות שונות.

```
SELECT    S-NAME, S-ADDRESS, ITEM-DESCRIPTION, PRICE
FROM      ORDERS, ITEM, SUPPLIER
WHERE     IT-NUMBER = A45B1   AND
          PRICE > 2500    AND
          S-NUMBER = SUP-NUMBER   AND
          IT-NUMBER = ITEM-NUMBER
ORDER BY S-NAME
```

— יש לקבל את רשימת הפריטים שבהם מופיעה בשדה תיאור פריט, המילה SCREW.
זוהי דוגמא לחיפוש מחרוזת תווים בתוך רצף של תווים.

```
SELECT    *
FROM      ITEM
WHERE     ITEM-DESCRIPTION LIKE %SCREW%
```

במערכת טבלאית אין קשרים מוגדרים מראש בין הנתונים, ולפיכך ההשוואה לשם בחינת הקשר בין שדות בטבלאות שונות יכולה לגזול זמן רב. כדי לשפר את הביצועים יש להגדיר אינדקסים לשדות. את האינדקסים יוצרים בתוכנית שירות לאחר טעינת הטבלה או לאחר עדכון.

3.2 אופרטורים קבוצתיים

אופרטור קבוצתי פועל על קבוצה שנבחרה לפי תנאי השליפה (WHERE) מתוך טבלה אחת או יותר. האופרטור פועל על שדה (עמודה) בקבוצה של השורות שנשלפו. השפה מכירה ב־5 אופרטורים:

COUNT — למנות מספר מופעים בתוך קבוצה מוגדרת (במקרה זה — מספר השורות שנשלפו).
SUM — לסכם תוכן שדה כלשהו.
AVG — לחשב ממוצע של שדה בתוך הקבוצה.
MAX — למצוא ערך מקסימלי של שדה בקבוצה.
MIN — למצוא ערך מינימלי של שדה כלשהו בקבוצת השורות שנשלפה.

אופרטורים אלו מגדילים את עוצמת השפה ותורמים לשיפור העבודה ולחסכון בזמן של המשתמשים. את האופרטורים הקבוצתיים רושמים במשפט SELECT. אם לא רושמים כל שדה שהוא, התוצאה שתוצג תכיל רק את התוצאה של פעולת האופרטור. אם יצויינו שמות שדות, הם יוצגו גם כן עם התוצאה של פעולת האופרטור. להלן מספר דוגמאות.

— המשתמש מעוניין לדעת את מספר הספקים המספקים את פריט A48.

```
SELECT   COUNT (DISTINCT SUP-NUMBER)
FROM     ORDERS
WHERE    ITEM-NUMBER = A48
```

DISTINCT מציין שהמונה יכיל את מספר המופעים של מספרי ספק שונים בלבד ולא יכיל מופעים כפולים של הזמנות מאותו ספק.

— יש לחשב את המחיר הממוצע של פריט A48 בכל ההזמנות שהוא מופיע.

```
SELECT   AVG (PRICE)
FROM     ORDERS
WHERE    ITEM-NUMBER = A48
```

— יש לסכם את הכמות המוזמנת מפריט A48 בכל ההזמנות שבהן מחירו נע בין 1000 ל-2000 שקל. יש להציג את תיאור הפריט, המחיר הממוצע בכל ההזמנות וסיכום הכמות שלו.

```
SELECT   ITEM-DESCRIPTION, AVG (PRICE), SUM (QUANTITY)
FROM     ORDERS, ITEM
WHERE    IT-NUMBER = A48 AND
         PRICE BETWEEN 1000 AND 2000 AND
         IT-NUMBER = ITEM-NUMBER
```

— המשתמש מעוניין לדעת מהו המחיר הגבוה ביותר המשולם עבור פריט A48 באיזו הזמנה שהיא.

```
SELECT   MAX (PRICE)
FROM     ORDERS
WHERE    IT-NUMBER = A48
```

אם הוא מעוניין גם במספר הספק, עליו לכתוב את ההוראה:

```
SELECT   SUP-NUMBER, MAX (PRICE)
FROM     ORDERS
WHERE    IT-NUMBER = A48
```

3.3 שאילתות מורכבות

במערכת SQL אפשר לבנות שאילתות מורכבות מאד תוך שימוש **בתת־שאילתות** (Subquery). הפלט של תת־השאילתא, שהינו למעשה טבלה, נמסר כקלט לשאילתא הראשית. נשתמש במספר דוגמאות כדי להציג את השימוש בטכניקה זו.

— המשתמש מעוניין לדעת את נתוני ההזמנה מבין ההזמנות של ספק 109 עבור פריט A55 שבה מחיר הפריט גבוה מהמחיר המקסימלי שבין ההזמנות של ספק 117 עבור אותו פריט. אילו לא ניתן היה להשתמש בתת־שאילתות היה צורך לפרק שאילתא זו לשתי שאילתות נפרדות. השאילתא הראשונה היתה מאתרת את המחיר המקסימלי עבור פריט A55 מבין כל ההזמנות של ספק 117. תוצאה זו היתה משמשת כקלט לשאילתא שניה שהיתה מחפשת מחיר גבוה יותר ו־ין כל ההזמנות של ספק 109. ב־SQL ניתן לשאול שאילתא מורכבת זו בצורה ישירה. את תת־השאילתא תוחמים בסוגרים כאילו היתה שם־שדה בתנאי השליפה.

```
SELECT  ★
FROM    ORDERS
WHERE   IT-NUMBER = A55 AND
        SUP-NUMBER = 109 AND
        PRICE >
        (SELECT MAX (PRICE)
        FROM ORDERS
        WHERE   IT-NUMBER = A55 AND
                SUP-NUMBER = 117)
```

— המשתמש מעוניין לדעת מיהו הספק המספק את הפריט B85 בכמות גדולה מהכמות הממוצעת המסופקת על ידי ספק 125, יש להציג את השדות מספר־ספק, שם־ספק, כתובת־ספק. אין להשתמש כאן בסימון ★ מכיון שפועלים על שתי טבלאות ואין כוונה להציג את השדות של שתיהן.

```
SELECT  SUP-NUMBER, S-NAME, S-ADDRESS
FROM    ORDERS, SUPPLIER
WHERE   IT-NUMBER = B85 AND
        SUP-NUMBER = S-NUMBER AND
        QUANTITY >
        (SELECT AVG (QUANTITY)
        FROM ORDERS
        WHERE  IT-NUMBER = B85 AND
               SUP-NUMBER=125)
```

3.4 עדכון בסיס הנתונים

עדכון בסיס הנתונים נעשה בטבלאות שכל עמודותיהן מוגדרות ולפיכך הוא פועל על שורות שלימות או על שדות בשורה. אפשר להוסיף או לבטל שורה ולעדכן תוכן שדה. כדי להוסיף עמודה או לבטל עמודה בטבלה משתמשים בהוראות אחרות. את השורות שיש לעדכן מאתרים במשפטי SELECT לפי תנאי שליפה (WHERE) כמו בשאילתות. נציג מספר דוגמאות לפקודות עדכון של בסיס הנתונים.

— המשתמש רוצה להוסיף פריט חדש בטבלת הפריטים ITEM. כאן אין צורך בתנאי שליפה כי אפשר להוסיף את השורה בסוף הטבלה, או במקום כלשהו פנוי. את ערכי השדות החדשים רושמים בפרמטר VALUES לפי סדר השדות בהתאמה.

```
INSERT   INTO ITEM (ITEM-NUMBER, ITEM-DESCRIPTION)
VALUES   ('A49', 'SCREW 14×15×35')
```

— המשתמש רוצה לשנות את תיאור פריט A49.

```
UPDATE  ITEM
SET     ITEM-DESCRIPTION = 'SCREW 14×15'
WHERE   ITEM-NUMBER = A49
```

אפשר לבצע עדכון קבוצתי בכל או בחלק מהשורות של הטבלה כאשר לא מציינים תנאי שליפה, או שתנאי השליפה מתייחסים ליותר משורה אחת. זאת נראה בדוגמא הבאה.

— המשתמש רוצה להגדיל ב־10% את מחירי כל הפריטים המסופקים על ידי ספק 109.

```
UPDATE  ORDERS
SET     PRICE = 1.1 × PRICE
WHERE   SUP-NUMBER = 109
```

בדוגמא זו באה לידי ביטוי אפשרות נוספת שלא הזכרנו קודם — ביצוע חישובים והקניית תוצאות חישוב לשדה כלשהו. אפשר גם להציג את התוצאה כפלט. הפעולות החישוביות יכולות להיות מורכבות כמקובל בפעולות עיבוד נתונים.

— המשתמש רוצה לבטל את רשומת ספק 154 מטבלת הספקים.

```
DELETE  SUPPLIER
WHERE   S-NUMBER = 154
```

4. בטיחות נתונים

מערכת SQL מספקת מספר אמצעים בעלי עוצמה לשם אבטחת הנתונים. החשובים שבהם: טבלאות לוגיות והרשאות.

4.1 טבלה לוגית – VIEW

המשתמש במערכת מעוניין לעיתים רק בחלק מהנתונים הכלולים בטבלאות השונות ולרוב הוא משתמש במספר טבלאות לשם הצגת השאילתות. בדרך המקובלת היה עליו להגדיר מדי פעם מחדש את אוסף הנתונים (טבלאות, שדות ותנאי שליפה) או ליצור לעצמו טבלה ממשית נפרדת. כדי להקל עליו מאפשרת המערכת להציג לו **תצפית** קבועה (VIEW) של הנתונים שנדרשים על ידו. הדבר נעשה באמצעות **טבלה לוגית** שאינה קיימת באופן ממשי בבסיס הנתונים אלא נבנית על פי הטבלאות הממשיות והמשתמש פועל בה כאילו היתה טבלה ממשית. שיטה זו מגדילה את אי התלות בנתונים ואת יעילות העבודה. מאידך, תורמת השיטה לבטיחות הנתונים בכך שהיא מאפשרת להציג למשתמש רק את הנתונים המותרים לו ומסתירה מפניו את השאר.

דוגמאות לבניית טבלאות לוגיות:

— מדור הזמנות חו"ל מטפל רק בהזמנות מספקים בחו"ל. לשם כך ניתן להגדיר עבורו טבלה לוגית אשר תכיל את כל ההזמנות של ספקים מחו"ל אשר מספריהם בתחום 301 עד 500. טבלה לוגית זו תשחרר את המדור מזיהוי ההזמנות שלו בכל שאילתא בצורה מפורשת והיא גם תמנע את הגישה של אנשי המדור להזמנות מספקים אחרים.

הפרמטרים הנילווים להגדרת VIEW מכילים את שם הטבלה ואת שמות השדות.

```
CREATE  VIEW HORDERS (H-SUPP-NUMBER, H-IT-NUMBER,
        H-PRICE, H-QUANTITY) AS
SELECT  *
FROM    ORDERS
WHERE   SUPP-NUMBER BETWEEN 301 AND 500
```

ההוראה CREATE יוצרת VIEW, או טבלה לוגית HORDERS שבסוגריים רשומים שמות העמודות שלה. בהוראה SELECT שולפים את כל העמודות או חלק מהעמודות של הטבלאות הרשומות בביטוי FROM. עמודה בטבלה החדשה יכולה גם להיות תוצאה של פעולה חשבונית בעמודה של טבלה ממשית, כפי שנראה בדוגמה אחרת. בביטוי WHERE מציינים את תנאי השליפה של הנתונים מתוך הטבלאות הממשיות כדי לבנות את הטבלה הלוגית.

אנשי מדור הזמנות חו"ל המשתמשים בטבלה לוגית זו יכולים לטפל בה כאילו היתה טבלה ממשית בבסיס הנתונים. שינויים בטבלה הממשית שממנה נבנית הטבלה הלוגית, כגון הוספת עמודות חדשות, שינוי סדר עמודות ועוד, אינם משפיעים על הטבלה הלוגית והמשתמש בה אינו מרגיש בשינוי.

— מדור הזמנות חו״ל מעוניין תמיד לדעת את ערך־ההזמנה שהוא המכפלה שבין מחיר הפריט לבין הכמות המוזמנת. ניתן להוסיף לטבלה הלוגית שדה "ערך־הזמנה" ובכך לשחרר את אנשי המדור מרישום דרישה חוזרת לחישוב השדה. מכיון שזוהי טבלה לוגית השדה לא קיים בטבלה ממשית כלשהי והוא מחושב בכל פעם שיוזכר שמו (במקרה זה H-VALUE).

```
CREATE   VIEW HORDERS (H-SUPP-NUMBER, H-IT-NUMBER,
         H-PRICE, H-QUANTITY, H-VALUE) AS
SELECT   SUPP-NUMBER, IT-NUMBER, PRICE, QUANTITY,
         PRICE × QUANTITY
FROM     ORDERS
WHERE    SUPP-NUMBER BETWEEN 301 AND 500
```

— מדור הזמנות חו״ל מעוניין לקבל בטבלה שלו גם את שם הספק ואת תיאור הפריט. למרות ששני שדות אלה מופיעים בטבלאות שונות ניתן לבנות טבלה לוגית שתכיל שדות אלה.

```
CREATE   VIEW  HORDERS (H-SUPP-NUMBER, H-SUPP-NAME, H-IT-NUMBER,
         H-IT-DESCRIPTION, H-PRICE, H-QUANTITY, H-VALUE) AS
SELECT   SUPP-NUMBER, S-NAME, IT-NUMBER, ITEM-DESCRIPTION,
         PRICE, QUANTITY, PRICE × QUANTITY
FROM     ORDERS, SUPPLIER, ITEM
WHERE    SUPP-NUMBER BETWEEN 301 AND 500 AND
         SUPP-NUMBER = S-NUMBER AND
         IT-NUMBER = ITEM-NUMBER
```

4.2 קביעת הרשאות

מערכת SQL מאפשרת לקבוע את הפעולות שמותר לכל משתמש לבצע בטבלאות הממשיות או הלוגיות. ולצורך עניין ההרשאות אין כל הבדל בין טבלה ממשית לבין טבלה לוגית. הרשאה למשתמש ניתנת באמצעות פקודת GRANT שהשימוש בה נעשה גם כן על פי הרשאה. ההרשאות במערכת נקבעות ומנוהלות על ידי מנהל בסיס הנתונים.

הרשאות ניתנות לטבלאות, לעמודות ולשורות (גם לפי ערכים), לפעולות ולהרשאות. להלן מספר דוגמאות.

— יש להרשות לכל המשתמשים גישה לצורך קריאה בלבד (SELECT) לטבלה ההזמנות מספקי חו״ל.

```
GRANT   SELECT
ON      HORDER
TO      PUBLIC
```

— במדור הזמנות חו"ל יש משתמש אחד, USER-1, אשר מורשה גם לעדכן את הטבלה ולהוסיף הזמנות חדשות.

```
GRANT   UPDATE, INSERT
ON      HORDER
TO      USER-1
```

— משתמש USER-2 מורשה לעדכן את מחיר הפריט בהזמנה.

```
GRANT   UPDATE (PRICE)
ON      ORDERS
TO      USER-2
```

— למשתמש USER-3 ולמשתמש USER-4 מותר לקרוא ולעדכן את טבלת הספקים וטבלת ההזמנות.

```
GRANT   SELECT, UPDATE
ON      SUPPLIER, ORDERS
TO      USER-3, USER-4
```

5. סיכום

מערכת SQL מיועדת לשימוש על-ידי משתמש הקצה חסר הידע במחשבים וגם על-ידי המשתמש המנוסה. אחת התכונות היחודיות שלה הוא הניווט האוטומטי בתוך בסיס הנתונים אשר מאפשר למשתמש להגדיר מה הוא רוצה ולהשאיר בידי המערכת את ביצוע העבודה. תכונה נוספות היא השפה האחידה לעבודה אינטראקטיבית ולשילוב עם שפות תכנות בתוכניות מקוונות ובאצווה. המודל הטבלאי תורם לפשטות המערכת מנקודת מבט המשתמש ומקל עליו את הבנת מבנה הנתונים.

בנוסף לפונקציות שהוצגו בקצרה בפרק זה יש למערכת SQL מרכיבים נוספים כגון גישור לשליפת נתונים ממערכת הירררכית DL/1, מרכיב להכנת דוחות, אוסף תוכניות שירות אינטרקטיביות ליצירה ותחזוקה של בסיס הנתונים.

המערכת כוללת מילון נתונים פנימי אשר מחזיק את תיאור הטבלאות והגדרת השדות המרכיבים אותן. המילון בנוי במבנה טבלאי והמשתמש יכול לפעול בו בהוראות SQL כפי שהוא פועל על טבלאות הנתונים שלו.

שפת השאילתות החדשה שהוכרזה על ידי חברת יבמ, QMF—Query Management Facility מרחיבה את אפשרויות השימוש בשפת SQL לשאילתות ולהכנת דו"חות וכוללת אמצעי עזר נוספים המסייעים למשתמש שאינו מנוסה ומקילים ומייעלים את עבודתו של המנוסה.

מערכות לניהול בסיסי נתונים
(DBMS)

רשימת סיכום

נספח זה מציג את רשימת המערכות המסחריות לניהול בסיסי נתונים. רשימה זו אינה שלימה אבל היא כוללת את רוב המערכות הקימות. הרשימה מבוססת על סקר שוק שבוצע על ידי הירחון DATAMATION ועל מידע שהיה זמין בעת כתיבת הספר.

יצרן	שם המערכת	מיועד למחשבים	מבנה נתונים	שפת תכנות	שפת שאילתות
Applied Data Research	DATACOM/DB	IBM 370, 303X DOS/VS, OS/VS	מהופך	COBOL וכל שפה התומכת ב־CALL	DATAQUERY
ASHTON-TATE	DBASE II	Z80, 8080, 8085	טבלאי	שפה עצמאית	
Bancohio corp.	DIRECT	Honeywell 600, 6000, 66 מערכת הפעלה GCOS	מהופך	COBOL	
Burroughs corp.	DMS II	Burroughs 1700—7800 מערכת הפעלה MCP	רשתי	COBOL, RPG, PL/1, ALGOL	יש
Cincom systems	TOTAL	IBM 370, SYS/3 DOS/VS, OS/VS Honeywell 200, 2000 CDC/Cyber DEC/PDP 11 Burroughs 2500 — 4800 NCR Century Univac	רשתי	COBOL, PL/1 FORTRAN Assembler RPG	T—ASK
Computer Corporation of America	MODEL 204 — DBMS	IBM 370, 303X 43XX OS/VS, DOS/VS		COBOL, FORTRAN, PL1, Assembler	יש
Control Data Corp.	DMS 170	Cyber 70, 170 CDC 6000 Scope, NOS	CODASYL רשתי והרחבות יחודיות	COBOL, FORTRAN, Algol, Assembler	Query-Update
CRI Inc.	CREATE 3000	HP 3000 MPE III	טבלאי	COBOL, BASIC, FORTRAN, SPL	יש
Cullinane Data Base Systems Inc.	IDMS	IBM370 Univac 70	CODASYL רשתי	COBOL, FORTRAN PL/1	יש
DEC	DBMS-10/20 DBMS-11	DEC PDP 10/20 PDP/11	CODASYL רשתי	COBOL, FORTRAN	אין
Hewlett-Packard Corp.	IMAGE	HP 1000—3000	רשתי	COBOL	יש
Honeywell Inc.	IDS II	H 400 H 60/6000	רשתי (ולפני DBTG	COBOL	יש
Honeywell Inc.	DM-IV	H 60	CODASYL רשתי	COBOL	
IBM Corp.	IMS	IBM 370 43XX, 30XX	הירכי	COBOL, FORTRAN PL/1 Assembler	אין
IBM Corp.	GIS/2	IBM 370 43XX, 30XX	אחזור נתונים מחולל דוחות	אין	יש

יצרן	שם המערכת	מיועד למחשבים	מבנה נתונים	שפת תכנות	שפת שאילתות
IBM Corp.	DL/1	IBM 370 43XX DOS/VSE	הירכי	COBOL, PL/1 RPG II Assembler	אין
IBM Corp	SQL/DS	IBM 370 43XX,30XX DOS/VSE,VM	טבלאי	COBOL, PL/1 FORTRAN	יש
IBM Corp.	DB2	IBM 370 43XX,30XX MVS	טבלאי	COBOL, PL/1	יש
IBM Corp	QMF	IBM 370 43XX,30XX VSE, VM, MVS	אחזור נתונים מחולל דוחות	באמצעות DB2,SQL	יש
Intel Corp.	SYSTEM 2000	IBM 370, 43XX CDC 70 ,170 Univac 1100	מהופך	COBOL FORTRAN PL/1, Assembler	QUEST
International Data Base systems Inc.	SEED	IBM/370, 43XX, 303X CDC 6000 Cyber DEC PDP 10 ,11	רשתי CODASYL	COBOL ,FORTRAN	HARVEST
Logica Inc.	RAPPORT	IBM 370 DEC VAX/11 PDP/11 Univac 1100 DG Nova Eclipse ICL 1900	טבלאי	FORTRAN	יש
Mathematica Products Inc.	RAMIS II	IBM 370	רשתי	COBOL, FORTRAN, PL1, Assembler	
Micro Data Base Systems Inc.	MDBS III	מיקרו מחשבים המבוססים על מעבד Z80 או 8080 מערכת הפעלה CP/M	רשתי		
Philips	PHOLAS	Philips 1000	רשתי CODASYL		אין
Prime Computer Inc.	DBMS	Prime	רשתי CODASYL	COBOL, FORTRAN	
Relational Software Inc.	ORACLE	IBM 370 DEC PDP	טבלאי	COBOL, PL/1 FORTRAN, C, PASCAL, Assembler	יש
Relational Technology	INGRES	DEC VAX	טבלאי	FORTRAN PASCAL	יש
Scientific Information Retrieval	SIR	IBM 370 CDC 6000 Cyber	רשתי		

יצרן	שם המערכת	מיועד למחשבים	מבנה נתונים	שפת תכנות	שפת שאילתות
Software AG.	ADABAS ADABAS-M	IBM 370 43XX,303X DEC PDP 11 VAX	מהופך, רשתי	COBOL, FORTRAN Assembler PL/1	ADASCRIPT NATURAL
Sperry Univac Division	DMS 1100	Univac 1100	CODASYL רשתי	COBOL, FORTRAN PL/1	יש
Tandem Computer Inc.	ENCOMPASS	TANDEM	טבלאי	COBOL, FORTRAN TAL	ENFORM

רשימה ביבליוגרפית

הרשימה הביבליוגרפית כוללת מגוון של ספרים, חוברות, תקופונים ומאמרים בנושאים שנדונו בספר זה.

בנוסף קיימת ספרות רבה בתחומים שונים המתפרסמת על ידי יצרני המחשבים ועל ידי בתי־תוכנה המפתחים מערכות תוכנה לניהול קבצים ובסיסי נתונים.

הקורא המעונין יוכל לבחור חומר נוסף הכולל גם תיאור מערכות חומרה ויחידות אחסון ותיאור יישומים והפתרון שניתן בהם לבעית אחסון נתונים.

1. אחיטוב ניב: "התפתחות הביקוש לבסיסי נתונים מצד המשתמשים", מעשה חושב, כרך ד׳, מס. 8, 1977.
2. ארגון וניהול קבצים, המכון לפריון העבודה והייצור.
3. בוסל יצחק: "הגישות השונות למערכות לניהול בסיסי נתונים", מעשה חושב, כרך ז׳, מס. 7, 1980.
★★4. הייפרמן רז: "ארגון נתונים וניהול קבצים", הוצאת הוד־עמי, 1982.
5. טוקר אבנר: "בסיסי נתונים", דפים לתלמיד בהוצאת המכון לפריון העבודה והייצור, 1980.
6. לדור מנחם: "עיצוב מסדי נתונים ב־IMS", יבמ ישראל, 1979.
7. עמיהוד יצחק: "תכנון ותכנות יישומי מחשבים", הוצאת הוד־עמי 1975.
8. עמיהוד יצחק, קורפל יונתן: "יסודות בניתוח ותכנון מערכות מידע", הוצאת הוד־עמי, 1980.
9. פרידמן מרק: "תעשית בסיסי הנתונים/מבט שונה", מעשה חושב, כרך ח, מס. 5 1981.
10. פרץ אברהם, שמואל דוד: "בחינת תוכנה לניהול בסיסי נתונים עבור המל"מ", מעשה חושב, כרך ו׳, מס. 3, 1979.
11. שי דוד: "קישור בין מסדי הנתונים", מעשה חושב, כרך ח׳, מס. 3. 1981.
12. שפיגלר ישראל: "מילוני נתונים", מעשה חושב, כרך ז׳, מס. 6, 1980.

★★ ספרי רקע ללימוד הנושא

★ ספרים מומלצים אחרים

1. Abrial J.R. [1974]: "Data Semantics" in Data Base Management editors Klimbic and Koffeman, North-Holland, Amsterdam.
2. ADABAS: "Introduction Manual", Sotware AG of North-America.
3. ADABAS: "Design Guide", Sofware AG of North-America.
4. Ahituv N., Hadass M. [1980]: "Identifying the need for a DBMS" Journal of Systems Management, Vol 31, No 8, pp 30.
5. Anderson J. P. [1972]: "Security in a Multi-user Computer Environment" in Advances in Computers, editor Rubinoff M., Academic Press.
6. ANSI/X3/SPARC [1975]: "Interim Report ANSI/X3/SPARC Study Group on Data Base Management Systems".
7. Armstrong W.W. [1974]: "Dependency Structures of data base relationships Proc. IFIP Congress, pp 580—583, North-Holland, Amsterdam.
8. Astrahan M.M. [1976]: "System R. Relational Approach to Data base Management, ACM Trans. Database Sys. Vol 1, pp 97.
9. Astrahan M.M., et all [1979]: "System R—A Relational Data Base Management System", Computer, Vol 12, NO 5, pp 211.
★10. Atre S. [1980]: "Data Base — Structured techniques for design, performance and management", John-Villey and Sons.
11. Auerbach [1975]: "Auerbach Guide to Data Base Management"Auerbach Publishers.
12. Bachman C.W. [1972]: "The Evolution of Storage Structures" CACM, Vol 15, No 7, pp 628.
13. Bachman C.W. [1973]: "The programmer as Navigator" CACM, Vol 16, pp 653.
14. Barnhardt R.S. [1980]: "Implementing Relational Data Bases" Datamation, October, pp 161.
15. Berztiss A.T. [1975]: "Data Structures, Theory and Practice" Academic Press.
16. Canning R.G. [1972]: "The Debate On Data Base Management" EDP Analyzer, 10, Vol 3.
★★17. Cardenas A.F. [1975]: "Analysis and Performance of Inverted Data Base Structures" CACM, Vol 18, No 5.
★18. Cardenas A.F. [1979]: "Data Base Management Systems" Allyn and Bacon Inc.
19. Champine G.A. [1979]: "Current trends in Data Base Systems" Computer, Vol 12, No 5, pp 196.
20. Chralovsky V. [1981]: "From files to Database — A tutorial" Information and Management, Vol 4, No 3, pp 115.
21. Codasyl DBTG [1971]: "CODASYL DBTG Report" ACM, New-York.
★22. Codd E.F. [1970]: "A Relational Model of Data for large Shared Data Banks", CACM, Vol 13, No 6, pp 377.
23. Codd E.F. (1971]: "Further Normalization of the Relational Model" in Data Base Systems, editor Rustin R., Prentice-Hall, pp 33.
24. Curtice R.M. [1975]: "Data Independence in Data Base Systems"

Datamation, April, pp 65.
25. Curtice R.M. [1977]: "Integrity in Data Base Systems" Datamation, May, pp 64.
26. Curtice R.M., Dieckman E.M. [1981]: "A Survey of Data Dictionaries" Datamation, March, pp 135.
27. Date C.J. [1977]: "An Introduction to Database Systems" Addison-Wesley Publishing Company, 2nd edition.
28. Davies G.B. [1974]: "Management Information Systems, Couceptual Foundations, Structure and Development" MeGraw-Hill series in Management Information Systems.
29. Dieckman M.E. [1981]: "Three Relational DBMS" Datamation, September, pp 137.
30. Engles R.W. [1972]: "A Tutorial on Data-Base Organizations" Annual Rev. Automat, Programming, Vol 7, pp 1.
31. Fitches S.D. [1977]: "The Adaptable Database Management System" Database Journal, Vol 7, No 2, pp 2.
32. Flynn R.L. [1974]: "A Brief History of Data Base Management" Datamation, Vol 20, No 8, pp 71.
33. Gosh S.P. (1977]: "Data Base Organization for Data Management" Academic Press.
34. Gotlieb C.C., Gotlieb L.R. [1978]: "Data Types and Structures" Prentice Hall, Englewood Cliffs, N.J.
35. Hoffer J.A. [1979]: "Data Base Integrity Controls" Journal of Systems Management, Vol 30, No 8, pp 38.
36. I BM [1971]: IMS/VS publications, DL/1 publications, SQL/DS, QBE
37. IDMS [1975]: IDMS publications, Cullinane Corp.
38. Inman W.M. [1981]: The Evolution of Data Design" Database Journal, Vol 11, No 3, pp 2.
39. King J. [1981]: "Is a Database Management System Required" Database Journal, Vol 11, No 4, pp 12.
40. Knuth D.E (1968]: "The Art of Computer Programing: Fundamental Algorithms" Addison-Wesley.
41. Knuth D.E [1973]: "The Art of Computer Programming: Sorting and Searching" Addison-Wesley.
42. Lefkovits D. [1969]: "File Structures for On-line Systems" Spartan Books, New-York.
43. Lyon J.K. [1976]: "The Database Administrator" Wiley — Interscience, New-York.
★★44. Martin J. [1977]: "Computer Data-Base Organization", Prentice-Hall.
45. Miller M. [1977]: "A Survey of distributed Database Management Systems", Information and Management, Vol 1, pp 234.
46. Nijseen G.M., Ed..:"Architerture and Models in Data Base Management Systems", North-Holland, Amsterdam.
47. Nijssen G.M. [1982]: "Databases of the Future: Design and Methodology", Database Journal, Vol 12, No 1, pp 2.

48. Olle T.W [1978]: "An Historical Perspective of the CODASYL Database Proposals"
Database Journal, Vol 8, No 4, pp 9.
49. Olle T.W [1979]: "The Relational Approach compared with The CODASYL Approach", Database Journal, Vol 9. No 3, pp 15.
50. Polk W.J. [1981]: "Managing the Very Large Database" Datamation, September, pp 115.
51. Potier D., Leblanc P. [1980]: "Analysis of Locking Policies in Database Management", CACM, vol 23, No 10, pp 584.
52. Scheuerman P. [1978]: "On the Design and Evaluation of Databases", Computer, Vol 11, No 2, pp 186.
53. Schubart R.F. [1974]: "Directions in Data Base Management Technology", Datamation, Vol 20, No 9, pp 48.
54. Senko M., et all [1973]: "Data Structures and Accessing in Data base Systems", IBM SYS. Journal, Vol 12, No 1, pp 30.
55. Smith D., Smith M. [1979]: "Relational Database Machines", Computer, Vol 12, No 3, pp 28.
56. TOTAL: TOTAL publications, Cincom Inc.
★★57. Tsichiritzis D.C., Lochovsky F.H. [1977]: "Data Base Management Systems", Academic Press Inc.
58. Tsichiritzis D.C., Lochovsky F.H. [1978]: "Designing The Data Base", Datamation, August, pp 147.
59. Uhvowczik P.P [1973]: "Data Dictionary/Directories" IBM Sys. Journal, Vol 12, No 4.
★60. Ulman J. [1980]: "Principles of Database Systems" Computer Science Press Inc.
61. Walsh M.E. [1978]: "Update on Data Dictionaries" Journal of Systems Management, Vol 29, No 8, pp 28.
62. Weldon J.L. [1981]: "Data Base Administration" Plenum Press, New-York.
63. Wood C., et oll [1980]: "Data base Security, requirements, policies and models", IBM SYS. Journal, Vol 19, No 2, pp 229.
64. Yao S.B. et all [1978]: "Data Base Systems", Computer, Vol 11, No 9, pp 171.
65. Zloof M. M. [1977]: "QBE a data base Language", IBM SYS. Journal, Vol 16, No 4, pp 321.

מילון מונחים ואינדקס

מילון המונחים הקצר המצורף מגיש לקורא את המונחים העיקריים שהוא נזקק להם בעבודתו, בנושא שבו דן הספר. הקבלת המונחים באנגלית ובעברית תְסייע לקורא בעת עיון בספרות העזר העניפה שרובה נכתבה באנגלית.

במילון משולב גם אינדקס, אשר נותן את מראי המקום העיקריים של הדיון בנושא או במושג.

סדר המיון הוא לפי המונח האנגלי המקובל.

– מהו בסיס נתונים – Data-Base ?
– מהי מערכת לניהול בסיסי נתונים – DBMS ?
– מהם ישות, קשר, מודל נתונים פיסי ומודל נתונים לוגי, סכימה ומילון־נתונים?
– מהם הכלים לאבטחת שלימות ובטיחות הנתונים?
– מהן מכונות בסיסי־נתונים?
– מהם התכונות והמאפיינים העיקריים של המערכות המסחריות?

בסיסי נתונים – עקרונות, מודלים ויישומים מציג את התיאוריה של המער־כות לניהול בסיסי־נתונים (DBMS), את המודלים הלוגיים העיקריים (הירֵרכי, רשתי וטבלאי), ומספר מערכות מסחריות הפועלות בשוק: IMS, IDMS, TOTAL, ADABAS, SQL/DS.

בסיסי נתונים – עקרונות, מודלים ויישומים נועד להכשיר מנתחי מערכות, מתכננים ומתכנתים העוסקים בפיתוח יישומים. הוא יסייע לאחראים לניהול בסיסי נתונים (D B A –– Data Base Administrators) במילוי תפקידם ויתן להם כלים להעריך ולבחור את המערכת אשר מתאימה ביותר לדרישות היישומים.

זהו המשך לספר **ארגון נתונים וניהול קבצים**, מאת המחבר.

הספר מותאם לתוכניות הלימודים למנתחי מערכות המופעלות באוניברסי־טאות, במכללות, בבתי־ספר מקצועיים ובקורסים נבחרים.

בסיום כל פרק הובאו שאלות לחזרה ולתרגול. בסוף הספר הובאו ביבליוגרפיה, מילון מונחים ואינדקס.

המחבר

רז הייפרמן, יליד 1949, תואר ראשון בסטטיסטיקה וכלכלה ותואר שני בחקר ביצועים מטעם האוניברסיטה העברית בירושלים. בוגר קורס מנתחי מערכות של המכון לפריון העבודה והייצור.
עובד בחברת "טכנולוגיה מתקדמת" כמנהל פרויקט ומנתח מערכות. עד שנת 1981 עבד במרכז שרותי המידע של משרד התקשורת בתפקידים של מנתח מערכות ומנהל תחום מערכות לוגיסטיות. התמחה בתכנון וביישום של מערכות מידע הפועלות בסביבה של בסיסי נתונים ותקשורת.

מרצה בנושאים של בסיסי נתונים וארגון קבצים במכון לפריון העבודה והייצור ובמכללות למחשבים. הרצה בקורסים של המדור ללימודי חוץ של הטכניון ושל משרד התקשורת.

ספרים נוספים בהוצאת "הוד–עמי"

1. **המחשב האלקטרוני ועיבוד הנתונים**, י' עמיהוד
2. **תכנון ותכנות יישומי מחשבים**, י' עמיהוד
3. **הקלט למחשב**, ס' מצפי
4. **יסודות בניתוח ותכנון מערכות מידע**, י' עמיהוד וי' קורפל
5. **מערכות תקשורת נתונים**, א' סקופ
6. **ארגון נתונים וניהול קבצים**, ר' הייפרמן
7. **מיקרו־מחשבים וה–6800 – חומרה ותוכנה**, א' הראל

* הספרים באישור משרד החינוך.